



VRE4EIC

**A Europe-wide Interoperable Virtual Research Environment
to Empower Multidisciplinary Research Communities
and Accelerate Innovation and Collaboration**

Deliverable D3.4

Enhanced VREs

VRE4EIC DELIVERABLE

Name, title and organisation of the scientific representative of the project's coordinator:

Mr Peter Kunz t: +33 4 92 38 50 10 f: +33 4 92 38 78 22 e: peter.kunz@ercim.eu

GEIE ERCIM, 2004, route des Lucioles, Sophia Antipolis, F-06410 Biot, France

Project website address: <http://www.vre4eic.eu/>

Project	
Grant Agreement number	676247
Project acronym:	VRE4EIC
Project title:	A Europe-wide Interoperable Virtual Research Environment to Empower Multidisciplinary Research Communities and Accelerate Innovation and Collaboration
Funding Scheme:	Research & Innovation Action (RIA)
Date of latest version of DoW against which the assessment will be made:	31 May 2017 Amended Grant Agreement through amendment n°AMD-676247-8
Document	
Period covered:	M1-33
Deliverable number:	D3.4
Deliverable title	Enhanced VREs
Contractual Date of Delivery:	30.06.2018
Actual Date of Delivery:	30.06.2018
Editor (s):	Peter Kunz
Author (s):	Carlo Meghini (CNR ISTI)
Reviewer (s):	Keith Jeffery (ERCIM) Cesare Concordia (CNR ISTI) Luca Trupiano (CNR ISTI) Theodore Patkos (FORTH ICS) Nikos Minadakis (FORTH ICS) Yannis Marketakis (FORTH ICS) Vangelis Kritsotakis (FORTH ICS) Daniele Bailo (INGV) Zhiming Zhao (UvA)
Participant(s):	All project partners
Work package no.:	3
Work package title:	Architecture, VRE development, integration and scalability
Work package leader:	Carlo Meghini (CNR)
Distribution:	PU
Version/Revision:	1.0
Draft/Final:	Final
Total number of pages (including cover):	59

What is VRE4EIC?

VRE4EIC develops a reference architecture and software components for VREs (Virtual Research Environments). This e-VRE bridges across existing e-RIs (e-Research Infrastructures) such as EPOS and ENVRI+, both represented in the project, themselves supported by e-Is (e-Infrastructures) such as GEANT, EUDAT, PRACE, EGI, OpenAIRE. The e-VRE provides a comfortable homogeneous interface for users by virtualising access to the heterogeneous datasets, software services, resources of the e-RIs and also provides collaboration/communication facilities for users to improve research communication. Finally it provides access to research management /administrative facilities so that the end-user has a complete research environment.

Disclaimer

This document contains description of the VRE4EIC project work and findings.

The authors of this document have taken any available measure in order for its content to be accurate, consistent and lawful. However, neither the project consortium as a whole nor the individual partners that implicitly or explicitly participated in the creation and publication of this document hold any responsibility for actions that might occur as a result of using its content.

This publication has been produced with the assistance of the European Union. The content of this publication is the sole responsibility of the VRE4EIC consortium and can in no way be taken to reflect the views of the European Union.

The European Union is established in accordance with the Treaty on European Union (Maastricht). There are currently 28 Member States of the Union. It is based on the European Communities and the Member States cooperation in the fields of Common Foreign and Security Policy and Justice and Home Affairs. The five main institutions of the European Union are the European Parliament, the Council of Ministers, the European Commission, the Court of Justice and the Court of Auditors (<http://europa.eu/>).

VRE4EIC has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 676247.

Table of Contents

1	Introduction	6
2	The VRE4EIC technical architecture	7
2.1	The e-VRE microservices design principles	8
3	Implementing the VRE4EIC technical architecture: the e-VRE prototype	8
3.1	The <i>choreography</i> approach in e-VRE: defining events and messages	9
3.2	The Node Service	11
3.2.1	The Node Manager	12
3.2.2	The User Manager	13
3.2.3	The Communication Bus	14
3.2.4	The Node Service: implementation choices and technologies adopted	15
3.2.5	The Node Service: source code, documentation and set up	17
3.3	The e-VRE AAI implementation in the canonical prototype	17
3.3.1	Security and Trust Components	19
3.3.2	The Two-Factor Authentication (2FA) in e-VRE prototype	21
3.3.3	The AAI: technologies adopted	22
3.4	The Metadata Service	22
3.4.1	The Metadata Service: implementation choices and technologies adopted	23
3.4.2	Metadata Service: the source code	23
3.5	The e-VRE Workflow Service	24
3.5.1	The Workflow Configurator	25
3.5.2	The GUI, the Workflow Executor, the Workflow Repository	26
3.5.3	Workflow Service Implementation description	28
3.5.4	Source code, documentation and set up	31
3.6	The e-VRE App Service	31
4	The VRE4EIC Graphical User Interface	32
4.1	Introduction to this section	32
4.2	Targeted Objectives of the Design	32
4.3	Functional Model	33
4.4	Architecture of the VRE4EIC GUI	40
4.4.1	Front End	41
4.4.2	Back End	42
4.5	Interactions of the VRE4EIC GUI with the e-VRE building blocks	43
4.5.1	Node Service	43
4.5.2	Metadata Service	43
4.6	Technologies used in implementing the VRE4EIC GUI	43
5	The Enhanced EPOS VRE	44
5.1	EPOS Integration within VRE4EIC	45
5.1.1	Provision of metadata describing EPOS assets	45
5.1.2	Provision of Scientific Background, use case and tools	45
5.1.3	EPOS integration within VRE4EIC - Workflows	46
5.2	EPOS enhancement by means of VRE4EIC building blocks	47
5.2.1	EPOS architecture enhancement	47
5.2.2	EPOS functionality enhancement	48
5.3	Conclusion of this section	50

6	The Enhanced ENVRIplus VRE.....	50
6.1	Community catalogue for cross-RI data and services	50
6.1.1	Short term: manually setting up a CERIF-based data catalogue	50
6.1.2	Long term: automatically harvesting CERIF records from diverse ENVRI RI catalogues	51
6.2	Cross-RI data and service discovery.....	52
6.3	Cross-RI workflow composition.....	53
6.4	Cross-infrastructure workflow execution and provenance	54
7	Conclusions	55
8	References	55
9	Annexes	56
9.1	Using 2FA in e-VRE prototype	56
9.2	VRE4EIC GUI implemented functionalities	57

1 Introduction

This deliverable provides the results of two tasks within WP3 of the VRE4EIC project: Task 3.4 “Integration of Reference VRE and Enhanced Existing VREs” and Task 3.5 “Design and development of a GUI for eVRE”.

In particular:

- The enhanced VREs are extended implementation of the EPOS and ENVRIplus VREs which have been obtained by migrating the architectures of these VREs into new architectures that includes building blocks developed by the VRE4EIC project, in the context of task 3.3. The resulting architectures provide the involved VREs with additional functionality that extends their capabilities in significant ways for the underlying research communities, while at the same time prove the functional and technical validity of the Reference Architecture developed (D3.1) and implemented by the project.
- The eVRE GUI provides eVRE with an access point for the human user, thus addressing the usability of VREs, which is one of the main goals of the project.

In addition, the present deliverable provides the Canonical Reference Prototype (CRP) of the Reference Architecture, also produced by Task 3.4 of the project by integrating existing technologies to implement the components of the Reference Architecture. The realization of the CRP has required the development of a Technical Architecture, which complements the Reference Architecture by selecting a set of suitable implementation techniques and open-source components.

The deliverable is structured as follows:

- Section 2 presents the VRE4EIC technical architecture and motivates the choices that inform it.
- Section 3 describes the CRP by first introducing the general approach followed in the implementation (the choreography approach) and then by presenting the implementation of each microservice in the Technical Architecture.
- Sections 4 presents the GUI of eVRE.
- Sections 5 and 6 describe the two enhanced VREs, EPOS and ENVRIplus, respectively.
- Section 7 concludes.
- The Annex reports useful information from deliverable 3.3: how 2-factor authentication works, and the complete list of functionalities implemented by the GUI, thereby establishing a link between the implementation of the CRP described in this deliverable and that of the building blocks described in D3.3.

It must be said that each of the four technical realizations presented in this deliverable is an outstanding achievement in its own right, especially with respect to the amount of resources used for them, time included. A key factor to these achievements has been the principled methodology followed throughout the project, consisting in the appropriate combination of the classical top-down approach in system engineering with the continuous interaction with the EPOS and the ENVRIplus users and developers, which has guided the VRE4EIC software architects and developers.

2 The VRE4EIC technical architecture

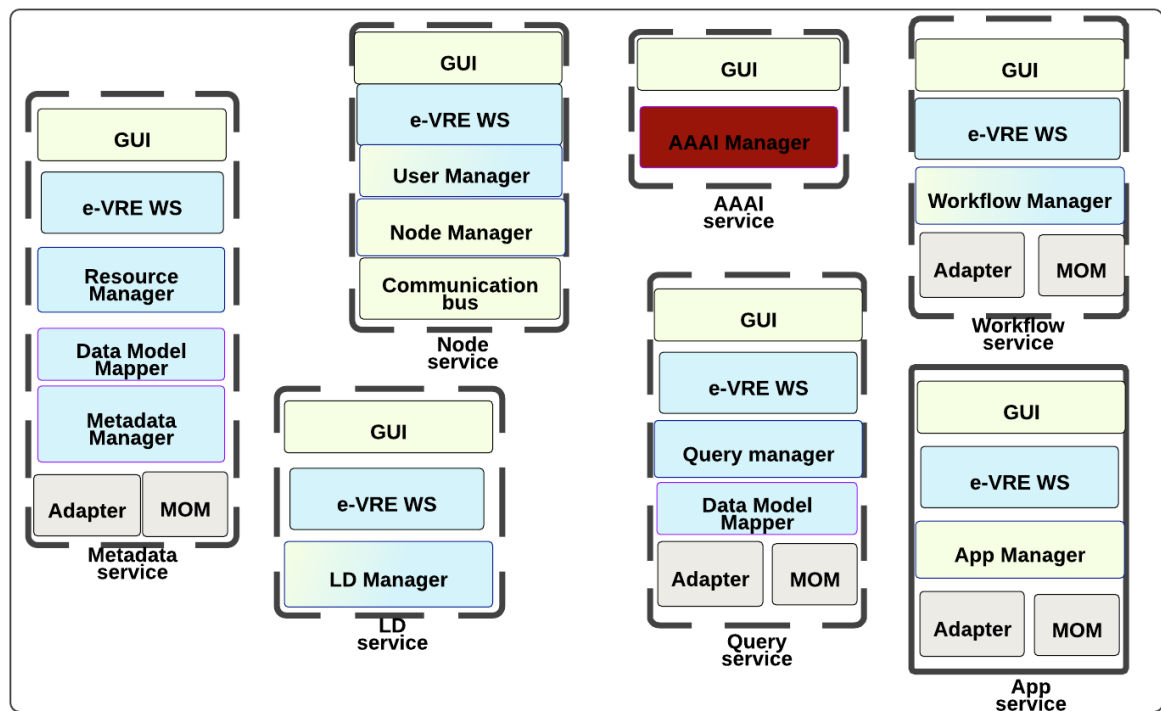
The key point in the derivation of the technical architecture, has been the VRE4EIC non-functional requirements defined in the project proposal:

1. The developed Virtual Research Environments must be a dynamic system: it should reuse and integrate *existing VRE tools, services, standardized building blocks and workflows where appropriate* [vre4eic], and develop *new innovative ones where needed*
2. The e-VRE should be *applicable to different multidisciplinary domains*, i.e. it can be potentially used in every research domain.
3. The e-VRE functionalities should be exposed as services in a standardized way to enable developers to easily use them to develop new applications.
4. The e-VRE must provide innovative standard software services to be retro-fitted to existing VREs to enhance them for their own domain purposes and for interoperability.

From the architectural point of view the above requirements mean that the e-VRE system must be *easily_expandable* (by adding or replacing software components), *highly_modular* (every building block should be independently deployable) and capable of supporting technology heterogeneity. We decided to adopt the Microservices approach for our technical architecture, since the two key concepts of Microservices architecture [Newman] perfectly fits the above requirements:

- *loose coupling*: every service knows as little as it needs to about the components with which it cooperates; this enables the Microservices to be independently deployable on existing VREs or replaceable in specific scientific domains
- *high cohesion*: components with related behavior stand together (i.e., related logic is kept in one service); changing the technology used to implement a Microservice does not affect other building blocks.

2.1 The e-VRE microservices design principles



We adopted a number of design principles to define the Microservices in our architecture:

- Use of asynchronous communications
- Maintain distributed process management in a single service by design
- Avoid service coupling because of component dependencies
- Efficiently manage integration with third-party software.
- Efficiently manage coexistence of different endpoints.

The Deliverable 3.3 [D 33] describes in details the solutions adopted at design level to implement these principles, in the next section an overall descriptions of the implementation of these principles is presented and the picture above shows which microservices has been created during this design.

3 Implementing the VRE4EIC technical architecture: the e-VRE prototype

This part of the document describes one possible implementation of the architecture designed in Deliverable 3.4. In the following sections are described: i) the overall approach followed to implement the functionalities of each Microservice defined in the technical architecture, ii) the technologies and standards adopted in each case and iii) the issues fixed and still opened.

In some cases, the overall instructions to install and run the microservices are described, however the complete installation guide for the e-VRE canonical prototype will be published on the VRE4EIC site. In the following will refer to e-VRE microservices as ‘building blocks’.

3.1 The *choreography* approach in e-VRE: defining events and messages

As described in Deliverable 3.3, the analysis of non-functional requirements has lead us to adopt *distribution of control* avoiding a central point of control in e-VRE. Our choice therefore went for the Choreography approach [D 33].

To implement this approach, an *event-driven* communication model has been adopted for interactions of building blocks. The first step for the implementation of this principle is the definition of the set of *events* that occurs in a system. According to [RUSS] “Events represent full, complete, self-describing and immutable *Facts* about the system”, and when creating a microservices architecture it is important to clearly define “which events should a service process” and “which events will a service emit”.



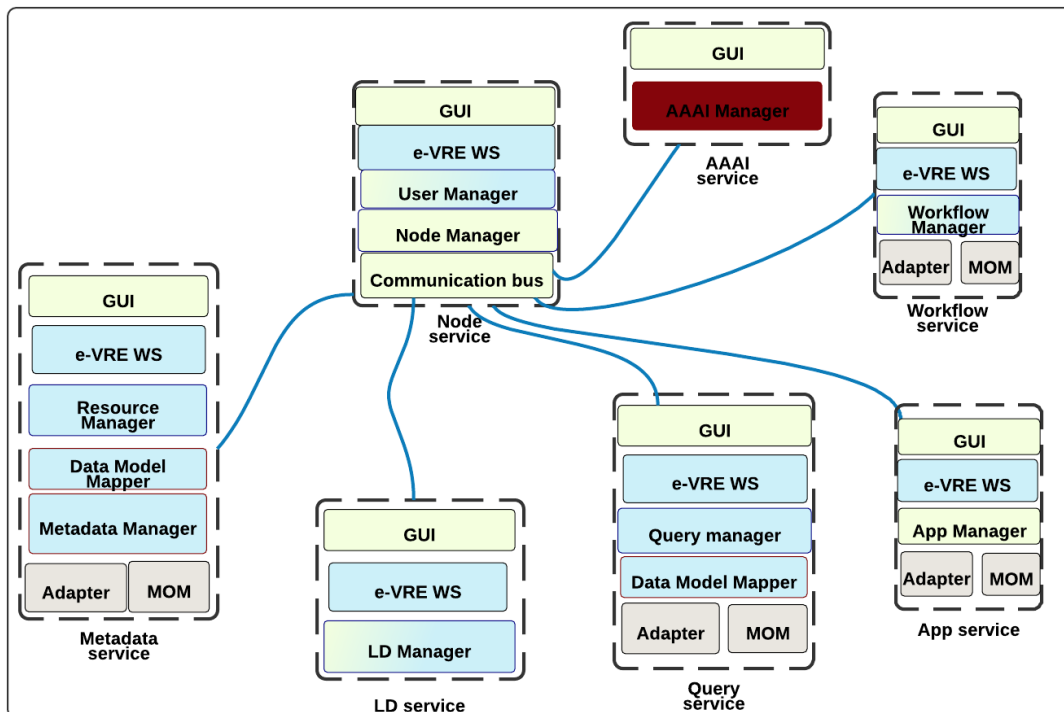
For every e-VRE microservice (building block) we have individuated the set of events it can emit by considering the list of the Generalized Functions (GF) extracted from the use cases [D 31], every completed GF produce a specific event emitted by the building block that has executed it; consequently, the building blocks that process the event will be those building blocks that could be affected by the GF executed. Since we tried to implement and *maintain the distributed process management in a single service* principle by design [D 33], most of the

events don't have any side effect to other building blocks and are emitted just to be processed for log reasons. However there are some special events that need to be processed by other building blocks: the most important of this events is the event generated by Fun21: Agent Authentication that must be processed by every other building blocks to guarantee the implementation of the Fun22: Continuous Access.

Events are communicated by microservices exchanging *messages*. To guarantee modularity, a microservice should not require any additional context, or dependencies on the in-memory session state to process a message representing an event; it must process the message and reacts accordingly if needed, without further information. The messages defined in e-VRE will be described in the section related to Node Service.

The event driven model in e-VRE is implemented using asynchronous communication. An asynchronous communication interaction is not blocking (the microservice initiating the communication does not wait for answer) and is highly decoupled (a producer of a message does not know who is going to react to its message). From an architectural point of view an event-driven interaction model reduces communication latency and improves scalability and flexibility of e-VRE (Requirements 1, 2, 4 above): for instance new publishers or subscribers can be added to (or can be removed from) processing an event message without other Microservices need to know it.

In e-VRE architecture the stream of events uses a Communication Bus that is implemented as a set of asynchronous communication channels, one for every category of events, managed by a specific component, the bus manager checks the producers/consumers credentials to permit a building block to execute or produce/consume operations.



Details of the Communication Bus implementation will be presented in the related sections.

3.2 The Node Service

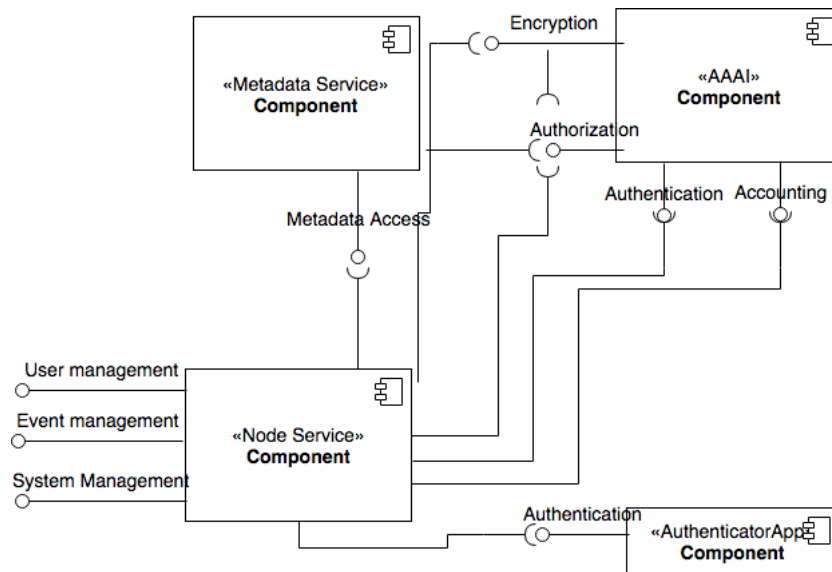
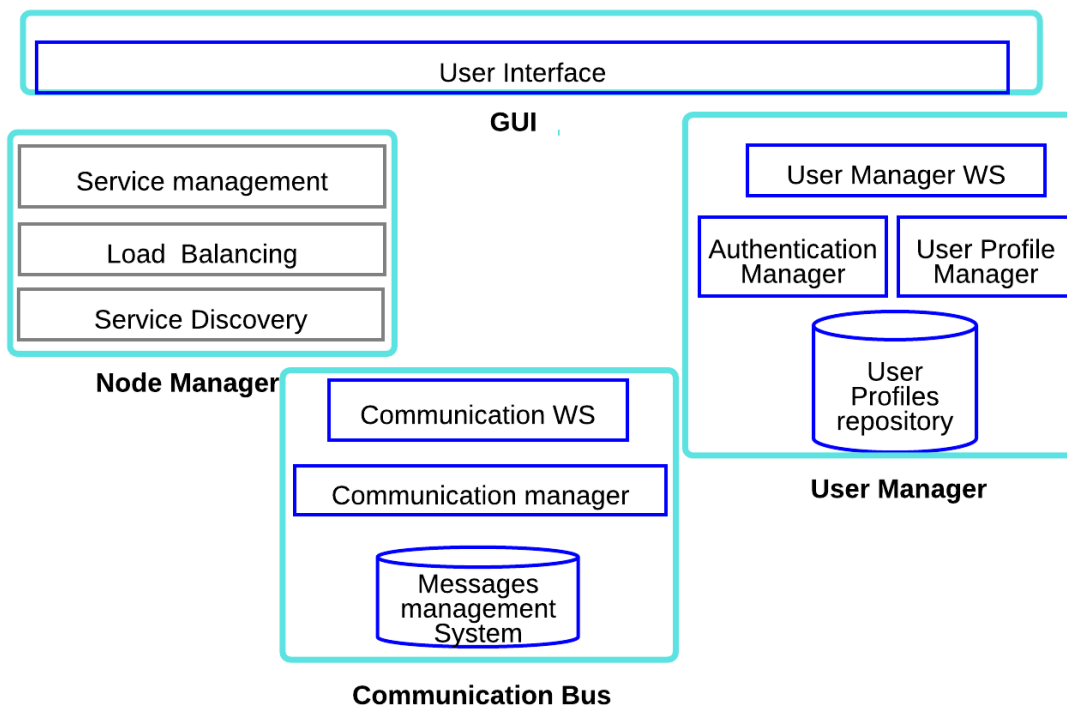


Figure 1 The Node Service UML Component Diagram

The Node Service implements all functionalities related to the User Profile management and e-VRE system administration. The figure 1 shows an UML component diagram which depicts the interaction with other services needed to implement its business logic.



The e-VRE Node Service functionalities are implemented by 4 main software components:

- a GUI that enables the administrator to monitor the system and define the configuration
- the User Manager that manages user profiles and provide authentication facilities
- the Communication Bus that is used as communication infrastructure for the e-VRE Microservices interaction.
- the Node Manager which implements the functionalities to manage the e-VRE system

3.2.1 The Node Manager

In a microservice system, services are typically distributed in a server infrastructure (created using containers and VM images). In such an infrastructure microservices may scale up and down based upon certain predefined conditions and the address of a microservice may not be known until the service is deployed and ready to be used.

The *Node Manager* implements the functionalities to manage a distributed configuration service, a synchronization service, and a naming registry.

Essentially, the Node Manager role is to enable the set of autonomous e-VRE microservices to act as a coherent single system.

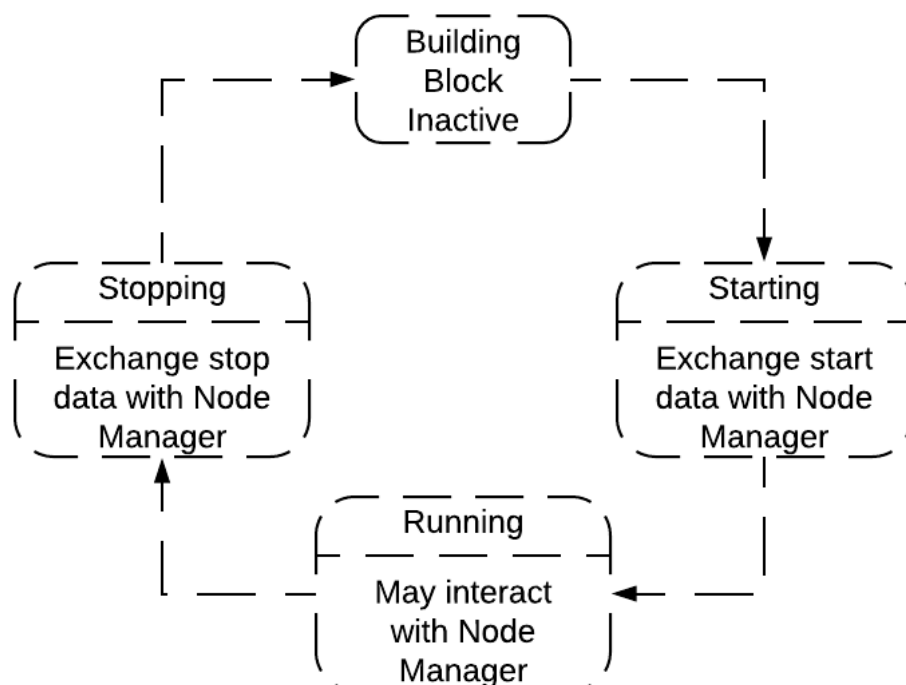


Figure 2 Partial state diagram of e-VRE building blocks life-cycle

Each e-VRE *building block*, knows the address of the Node Service and during the start phase of its life-cycle registers with the Node Manager component. During the registration, the building block:

- provides information about itself, such as the endpoint address.
- gets information it may need to execute its business logic: the address of other building blocks, credentials to access remote resources, certificates to implement encrypted communications etc.

At runtime, a building block can interact with the Node Manager, for instance to find out the location of other e-VRE building blocks or to communicate significant changes in its state. When the building block stops, it communicates with Node Manager to inform that it will be no longer available.

The Node Manager stores locally the information about the status of the e-VRE system and implements a policy for load balancing.

According to our architecture design, an *e-VRE system* is made by the set of running building blocks coordinated by a specific Node Manager.

3.2.2 The User Manager

The *User Manager* building block implements the management of User profiles containing information about the users that registers on e-VRE and *wraps* the authentication functionalities of the AAI building block.



Figure 3 The User Manager class

The functionalities of the User Manager building block are mainly used by external agents to register/authenticate in the e-VRE systems.

user-controller : User Controller

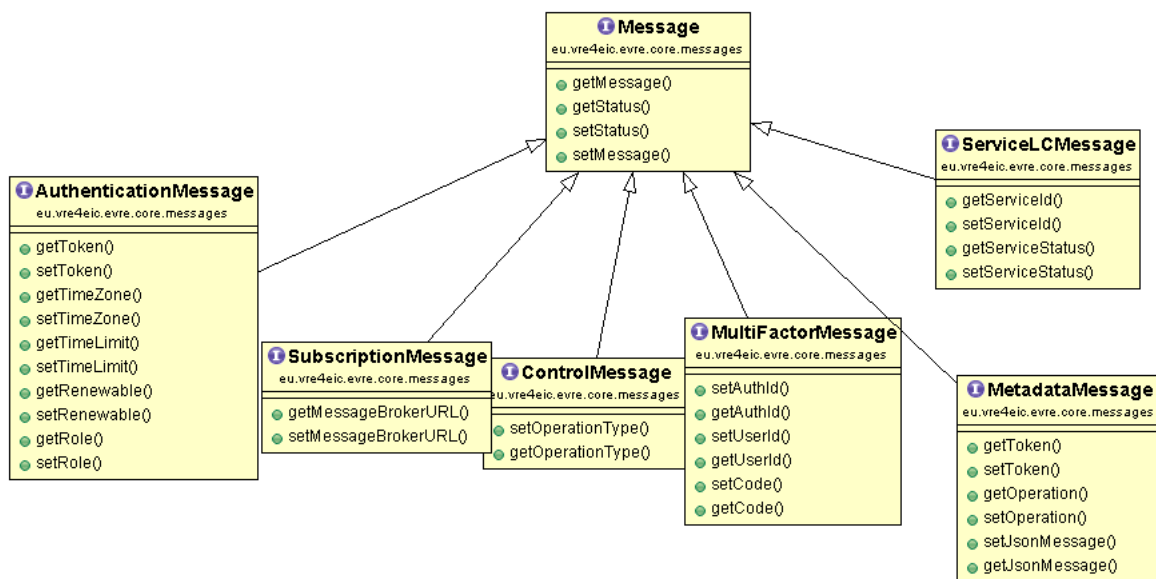
Show/Hide | List Operations | Expand Operations

GET	/user/checkevent	Checks the status of an e-VRE event
GET	/user/checkevents	Checks the status of all subscribed e-VRE events
POST	/user/createprofile	Creates a user profile on e-VRE
GET	/user/getevents	Returns the list of e-VRE events
GET	/user/getprofile	Gets a user profile based on user id
GET	/user/getprofiles	Gets a list of user profiles
GET	/user/login	Authenticates a user
GET	/user/loginmfa	Authenticates a user using a 2FA procedure
GET	/user/loginmfa	Invoked to complete the <i>Two-factor authentication</i> procedure started by a user
GET	/user/logout	Sign off a user
GET	/user/removemyprofile	Removes the profile of a user from e-VRE
GET	/user/removeprofile	An administrator removes the profile of a user from e-VRE
GET	/user/subscribeevents	Subscribes to a list of e-VRE events
GET	/user/unsubscribeevents	Unsubscribes from a list of e-VRE events
POST	/user/updateprofile	Updates the information in a profile of a user

Figure 4 Web Services entry points for e-VRE User Manager

3.2.3 The Communication Bus

The Communication Bus is responsible for managing the asynchronous interactions implementing the event driven model in e-VRE. It acts as a Message Oriented Middleware (MOM) and provides to e-VRE building blocks functionalities to exchange messages.



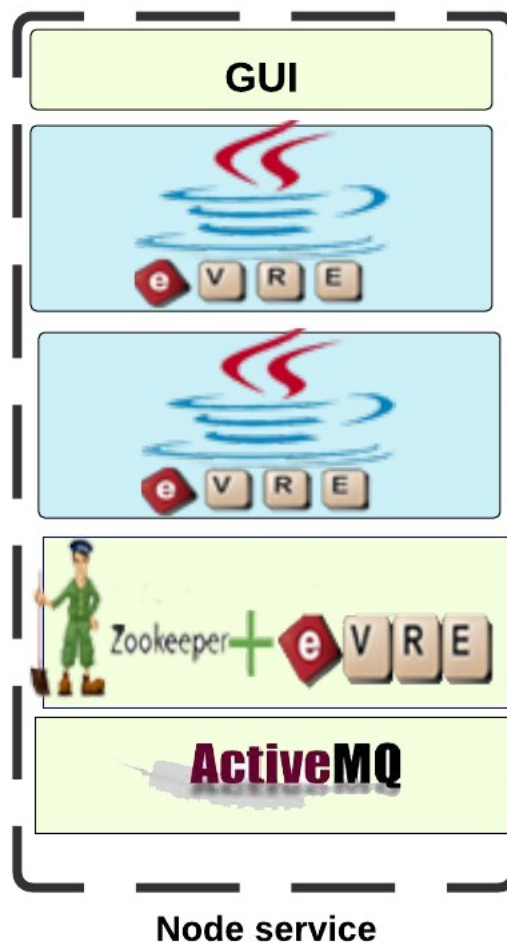
Class diagram of a subset of e-VRE Messages

Messages describe events that are relevant for the implementation of the business logic of the system, the image above shows a subset of the messages implemented in e-VRE.

Following the MOM principles, the e-VRE Communication Bus is used to create a number of communication channels, each one containing a specific kind of messages.

When a building block starts, it gets information about the system configuration from the Node Manager, including the list of active communication channels. It may *subscribe* to one or more channels, and may also ask the Communication Bus to create some channels. During its life-cycle the building block will be able to produce and consume (according to its permissions) messages on the channels it has subscribed.

3.2.4 The Node Service: implementation choices and technologies adopted



The *e-VRE Web Services* component has been implemented in Java, the entry points of this component are described and documented in details in the prototype site.

The same approach has been adopted to develop the *User Manager* component, a Java library has been built using MongoDB as persistence layer.

To implement the *Node Manager* in the e-VRE prototype the Apache ZooKeeper framework¹ has been used. Zookeeper provides functionalities to maintain status type information in memory and keeps a copy of the state of the entire system and persists this information in local log files. In e-VRE, every building block creates a znode (a file that persists in memory on the Node Manager server). The znode can be updated by building blocks that have permissions to do it, and any other building blocks in the e-VRE can register with the Node Manager to be informed of changes to that znode (i.e. to “watch” a specific znode).

In order to help developers to implement the logic described for an e-VRE building block we have created an helper class (a sort of Node Manager client), called NodeLinker class. The current release of this class is shown in the image below:

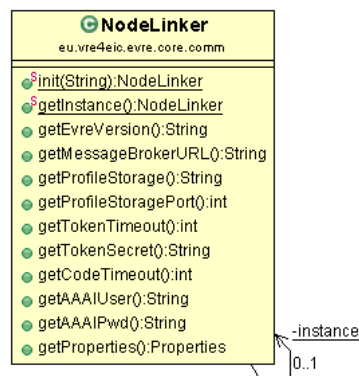


Figure 5 The NodeLinker class

The Node Linker class needs to be initialized with the address of the Node Manager and the credentials, when instantiated it automatically sends to the Node Manager the address and the name of the building block and downloads a number of properties that can be used by the building block.

This class will be upgraded in the next release to include also the exchange of security certificates.

Finally, the *Communication Bus* in the e-VRE prototype is based on Apache ActiveMQ², a framework implementing MOM principles; it is used as communication layer by a Java API developed by VRE4EIC team. In particular the code developed for Communication Bus add to ActiveMQ a security layer to encrypt and digitally sign messages exchanged, details of this security layer have been reported in the Deliverable 5.4 of the VRE4EIC project and will be also explained in the section related to AAAI building block.

¹ <https://zookeeper.apache.org>

² <http://activemq.apache.org>

3.2.5 The Node Service: source code, documentation and set up

The java code is published on GitHub, and can be downloaded at the following URL:

<https://GitHub.com/vre4eic/NodeService>

The Node Service has been developed as a Java Maven project, the code is separate in two main packages:

- *eu.vre4eic.evre.nodeservice* where there is the code that implements the functionalities of the Node Service building block.
- *eu.vre4eic.evre.core* where we implement the code implementing functionalities commons to all building blocks. In particular this package contains the Java classes implementing messages and the classes implementing Node Manager clients. This development choice has been taken in order to create a common API that can be used by all building blocks when interacting with each other or with the infrastructure. These API are distributed as Java archive (jar)

The java classes are documented in details using Javadoc, the complete documentation can be read here:

<http://v4e-lab.isti.cnr.it:8080/NodeService/doc/index.html>

For the documentation of e-VRE Web Services we adopted Swagger, a software framework that enables developer to describe the API. The swagger documentation of the e-VRE Web Services is here:

<http://v4e-lab.isti.cnr.it:8080/NodeService/swagger-ui.html#>

To set up the Node Service MongoDB and ActiveMQ are required in your environment.

At the time of the release of this deliverable the only way to install a Node Service is to clone or download the source code, then manually change property values in the file:

[your_dir]/NodeManager/src/main/resources/Settings.properties

In particular the MongoDB and ActiveMQ address and credentials must be set. When the file *Settings.properties* has been updated a Web ARchive (WAR) must be created using maven and deployed on an application container.

Before the end of the project we will create a distribution for e-VRE.

3.3 The e-VRE AAI implementation in the canonical prototype

To implement the functionalities related to authorization/authentication in e-VRE prototype we have delegate to the User Manager the role of partial wrapper of the AAI functionalities; this implementation pattern enables e-VRE prototype to be independent from the framework used to implement the AAI. In particular the User Manager wraps the interactions between the AAI framework and the external agents to execute two crucial operations: i) store credentials of user profiles created in e-VRE prototype and ii) check credentials validity when a user logs in the system.

As explained in Deliverable 3.3, in order to improve security we decided to implement here a synchronous communication: the User Manager interacts with AAI using a synchronous, encrypted channel.

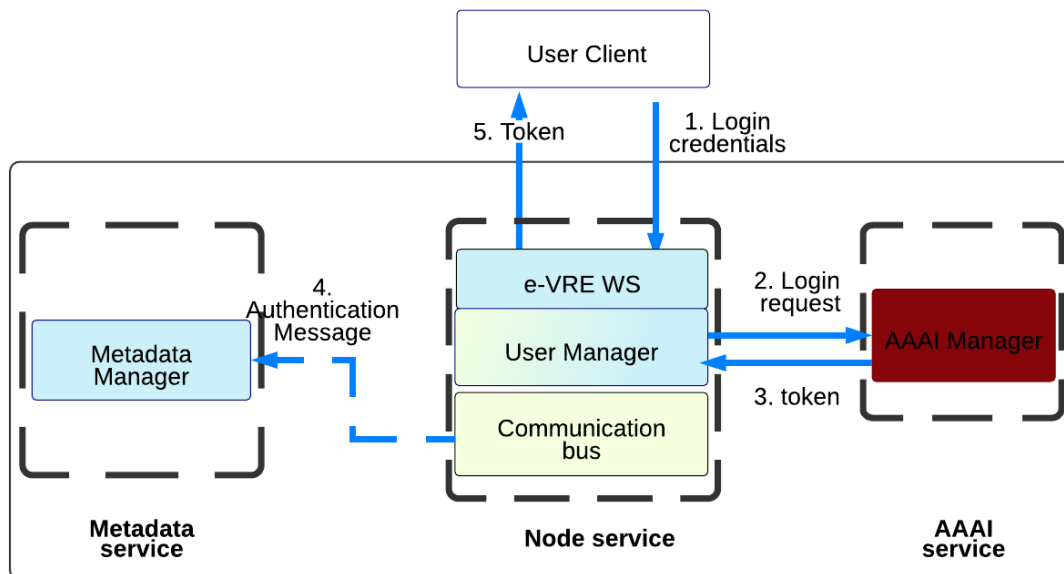


Figure 6 Login in the system using User Manager

The image above shows how authentication occurs and identity propagates when an external agent authenticates using the User Manager:

1. The client sends the credentials using a e-VRE WS entry point
2. the User Manager forwards the credentials to AAI and waits for answer
3. the AAI verify that credentials are valid and returns a *token*
4. the User Manager creates an e-VRE Authentication Message (explained in next sections of this deliverable) that contains the token and sends this message asynchronously via the Communication Bus
5. the User manager sends an answer to the client containing the token.

The client will use the token in every interaction with e-VRE building blocks for instance when executing queries on the catalogue via the Metadata Service. The figure 6 shows the UML *sequence diagram* describing this use case.

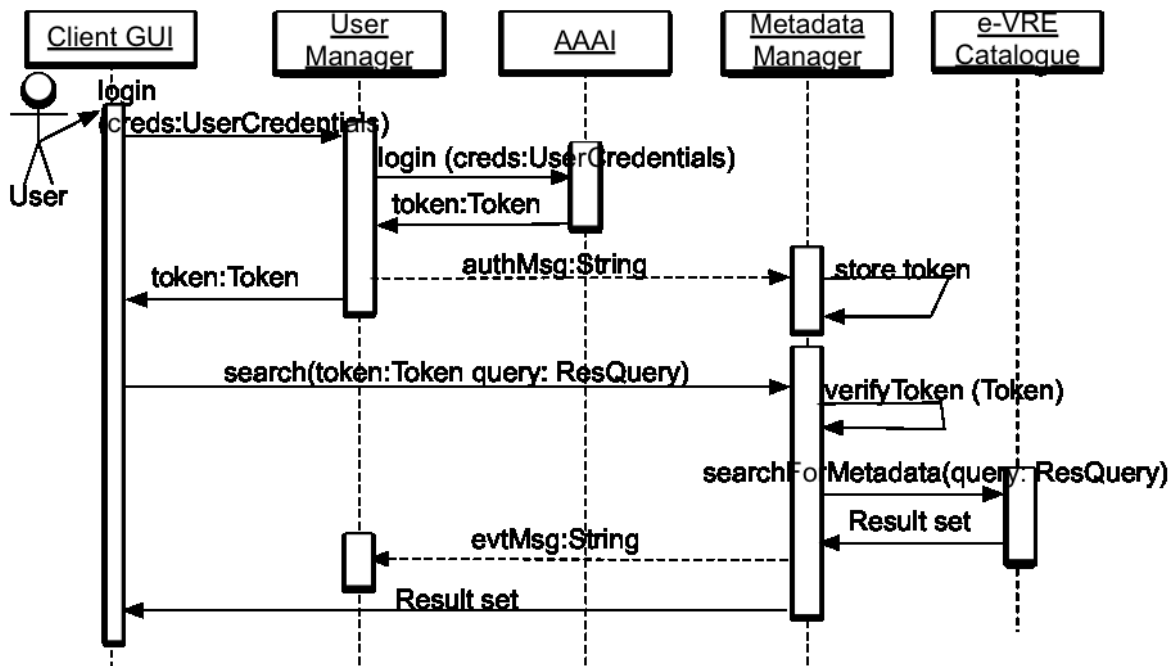


Figure 7 Login on e-VRE prototype and execute query

The main security/trust issue in this use case is that the Metadata Manager needs to know that the *token* it receives asynchronously has been created by an e-VRE building block that has the authority of creating it, and that it has not been tampered with. The solution adopted to solve this issue has been to sign and encrypt tokens and messages exchanged by e-VRE building blocks. In the use case the AAAI creates a token, encrypts it, and then returns it to the Node Service (as explained this happens on a secure encrypted channel), the Node Service creates an *AuthenticationMessage*, signs it and send the message asynchronously to e-VRE building blocks. When the Metadata Service receives the token: checks the signature, if it is a valid signature, decrypts the token and stores it locally.

Every e-VRE building block signs messages before publishing them in the Communication Bus channels and subscribers validate messages signature before consuming them.

To implement token encryptions and message signing, we have used the JSON Web Token (JWT) standard³. In the e-VRE prototype, JWT encryption and signature are based on shared private keys, exchanged by building blocks via Node Manager, a more secure (and efficient) Public Key Infrastructure will be implemented in next release.

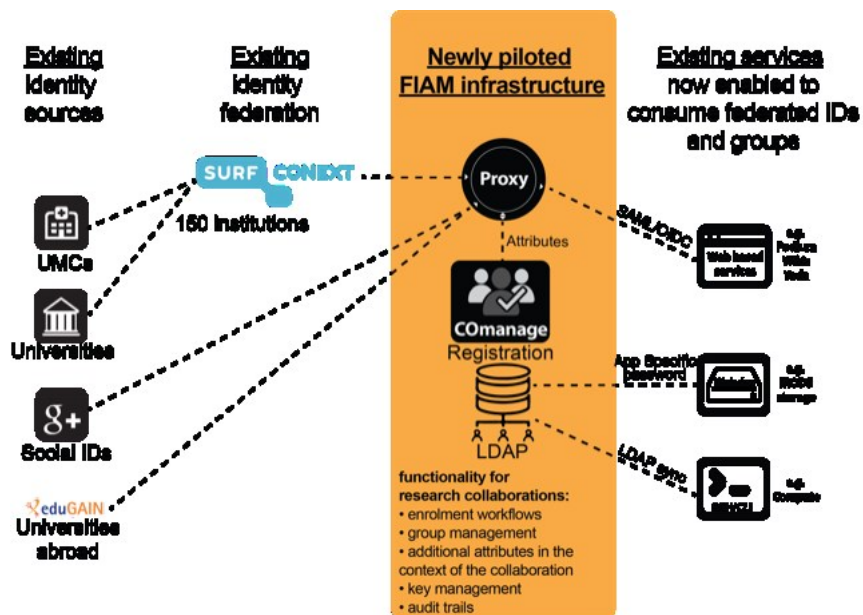
3.3.1 Security and Trust Components

Some key issues discussed in Deliverable D5.3 and D5.4 around Security and Trust require solutions that are either too domain- or platform-specific, or insufficiently standardized to be incorporated in the reference architecture or the canonical reference prototype. As a use case

³ The JWT is “an open standard [...] that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA”. Info: <https://jwt.io/introduction/>

in dealing with security and trust issues in a digital humanities use case, we explored user interface components to support a collaborative, explorative and interactive web front-end that can still produce transparent and reproducible results on (privacy) sensitive data. These components are also used to gain practical experience in connecting e-VREs to other (inter)federated authorization infrastructures that currently under development by the VRE4EIC project participating in a pilot⁴ managed by the Dutch National Research and Education Network (NREN) organization SURFnet.

Federated Authentication and Authorization Pilot



The VRE4EIC project participation in the pilot is to gain practical experiences with existing international standards for federated authentication (e.g. identity management) and (inter) federations such as eduGAIN, and to explore the landscape around federated authorization that is currently developing in projects such as AARC2. We collaborate since May 2018 in a specific pilot⁵ by SURFnet where several services are bundled in a virtual Science Collaboration Zone, and are particularly interested in the federated attribute management functionality. It allows us to explore the distributed management of user attributes stored at the Identity Provider, the federated infrastructure and/or the eVRE AAAI service. The Pilot has just started and a first working prototype has been demonstrated at an internal project meeting. We will fully report on this in deliverable D3.5

⁴ <https://wiki.surfnet.nl/display/SCZ/Pilot+partners>

⁵ Image taken from:

<https://wiki.surfnet.nl/download/attachments/57183089/Science%20Collab%20Zone%20simplified.png?version=2&modificationDate=1524202958775&api=v2>

3.3.1.1 *Collaborative executable notebook for transparent data science*

To demonstrate trust-specific components a public demonstrator has been developed⁶ and made available as a web service. This web service demonstrates the use of the SWISH datalab as a prototype of a transparent and collaborative executable notebook user interface for e-RIs and e-VREs platforms. The demo contains all the code necessary to reproduce the tables, figures and other computational results of an open access Web Science 2018 paper: “Using the Web of Data to Study Gender Differences in Online Knowledge Sources: the Case of the European Parliament”. It uses a collaborative notebook interfaces integrating the complete statistical analysis code (in R) for all results presented in the paper, and the complete declarative data pre-processing and data modeling code (in Prolog). This means that all computational steps leading to the resulted on in the paper are open for inspection and review. In addition, it offers permalinks for all interactive results obtained from the system, which means that all final and intermediary steps can be downloaded in the future, even if the underlying code in the collaborative notebook is changed by the researcher or her colleagues.

3.3.1.2 *Technical details of the release*

All software and code needed to reproduce the public web demonstrator has been archived for long term storage at Zenodo under DOI <https://doi.org/10.5281/zenodo.1232929>. To project this code against dependencies on operating-specific details and changes in third-party software, a self-contained virtual machine image has been published for long term storage at <https://doi.org/10.5281/zenodo.1237673>. All source code of the software is available on the VRE4EIC GitHub account (<https://GitHub.com/vre4eic/websci2018-reproducibility-pack>) and as Docker containers (<https://hub.docker.com/u/vre4eic/>).

3.3.2 **The Two-Factor Authentication (2FA) in e-VRE prototype**

The e-VRE prototype User Manager implements a two-factor authentication (2FA) method i.e. a method using a combination of two different factors to authenticate a user. The following sequence diagram shows the building blocks involved to implement this functionality.

⁶ <http://vre4eic.project.cwi.nl/gender/>

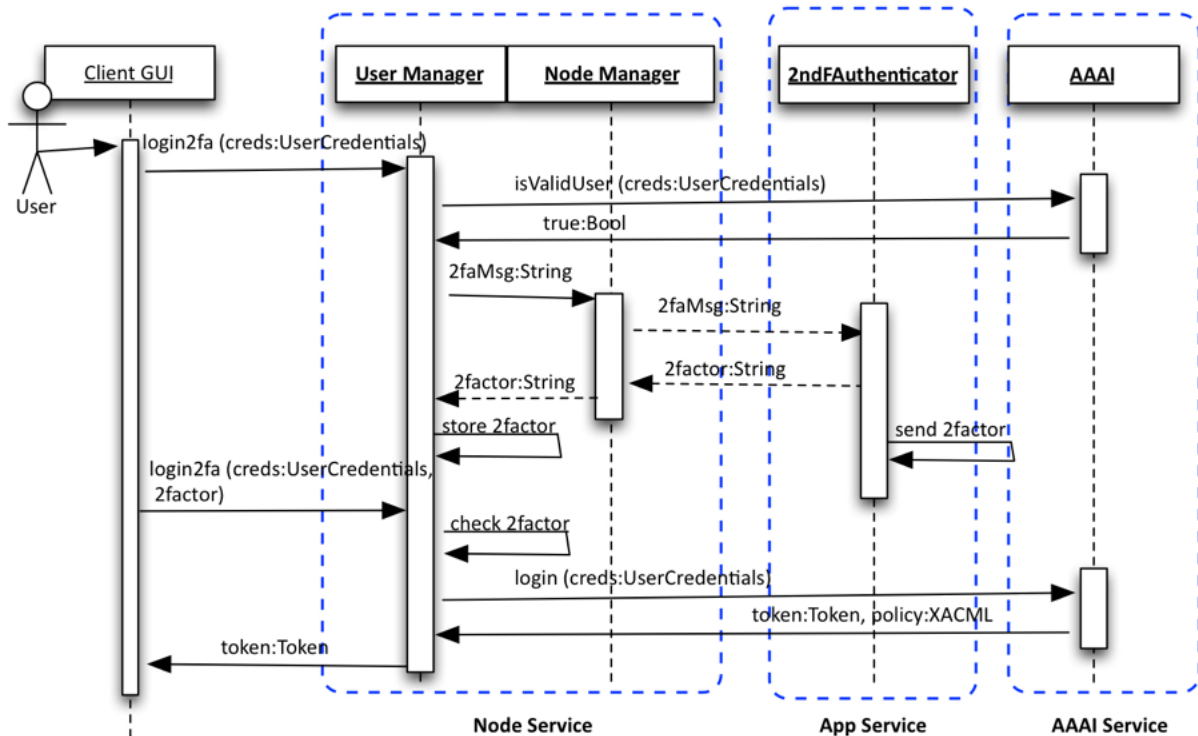


Figure 8 2FA in e-VRE prototype

In e-VRE prototype the channel used to communicate to the user the second *factor* is a Telegram Channel, a detailed description of how a user can use the 2FA feature to authenticate on e-VRE prototype is in VRE4EIC Deliverable 3.3 [D 33], in particular in sections 6.1 and 5.4.

The component integrating the telegram Framework is called *evre-TGBotAuthenticator*, the source code of this component is available here:

<https://GitHub.com/vre4eic/TelegramBots>

3.3.3 The AAAI: technologies adopted

The Unity IDM⁷ is used to implement AAAI functionalities for federated identity in the e-VRE prototype.

3.4 The Metadata Service

It is the e-VRE building block responsible for storing and managing resource catalogues. These functionalities are provided by exploiting various components (i.e. MetadataManager, DataModelMapper, etc.).

⁷ <http://www.unity-idm.eu/>

3.4.1 The Metadata Service: implementation choices and technologies adopted

During the design phase of the project it has been decided that this building block should have used a triple-store as repository. Initially, Blazegraph⁸ was installed as the Metadata Service repository. However, during the e-VRE prototype development, it was noticed that in many cases Blazegraph had performance issues. In addition to that, Blazegraph's development itself, seems to be inactive for the last couple years. For those reasons it was decided to seek for an alternative triple-store, ending up with adopting the Virtuoso Universal Server⁹. The Virtuoso Universal Server merges the capabilities offered by a hybrid database engine and by a middleware that combines the functionality of a traditional RDBMS, ORDBMS, RDFStore, virtual database, web application server and file server. It is in fact a single threaded server process that supports multiple protocols.

The following standard Web and Internet protocols have been implemented in Virtuoso: HTTP, HTTPS, WebDav, SOAP, UDDI, SPARQL and SPARUL. In addition, concerning the development of database-based applications and the integration of systems, Virtuoso has implemented a wide variety of industry standard data access APIs, such as ODBC, JDBC, OLE DB and ADO .NET. Virtuoso is made up of various server and client components, which enable the communication of a local or remote Virtuoso server, such as the Conductor, which is Web based Database Administration User Interface, and ISQL and ISQLO utilities. To this end, the Virtuoso Universal Server can be exploited to produce a clustered-based system of RDFStores. In addition, there is a specific version of this server, which can be deployed on the Amazon Cloud. Concerning the specific characteristics of Virtuoso that are of interest for e-VRE prototype, it can be highlighted that Virtuoso not only allows the management of RDF (linked-data) but it enables their querying through the SPARQL language. The same language (actually SPARUL) can be exploited for the updating of the linked-data.

3.4.2 Metadata Service: the source code

MetadataService has been developed as a JAVA maven project. It consists of different components, therefore it is important to separate the development spaces for these components. All the different components are placed under the eu.vre4eic.evre package. After that follows another package that groups together the resources of a particular component (i.e. eu.vre4eic.evre.metadatamanager contains all the resources of the corresponding component).

For each component we use different packages for grouping together resources that are used for a particular reason. More specifically we use the following packages:

- api: for adding the interfaces of the components. These are the contractual interfaces that are also visible from other components.
- impl: for adding the actual implementation (or different implementations) of the interfaces that are found under the api package.
- model: for adding the structural components that are required (i.e. POJOs)
- exceptions: for adding the corresponding exceptions-related resources

⁸ <https://www.blazegraph.com>

⁹ <https://virtuoso.openlinksw.com/universal-server/>

- test: for adding the resources that are needed for testing the components

Furthermore for grouping the resources that are exploited throughout all the components we have created a commons package (found under eu.vre4eic.evre.commons)

The Javadoc of the code developed can be found here:

<http://139.91.183.70/apidocs/>

The swagger documentation of the e-VRE Web Services is here:

<https://app.swaggerhub.com/apis/rousakis/ld-services/1.0.0>

3.5 The e-VRE Workflow Service

In the e-VRE Reference Architecture, the Workflow Manager (WM) component is responsible for the management of business processes and scientific workflows. To do this, the WM interacts with other components, for instance it uses the Metadata Manager to get information to build workflows and to store them, and the Query Manager to execute distributed queries. The figure below shows the UML component diagram describing the complete set of interactions of Workflow Manager with the components of the Reference Architecture.

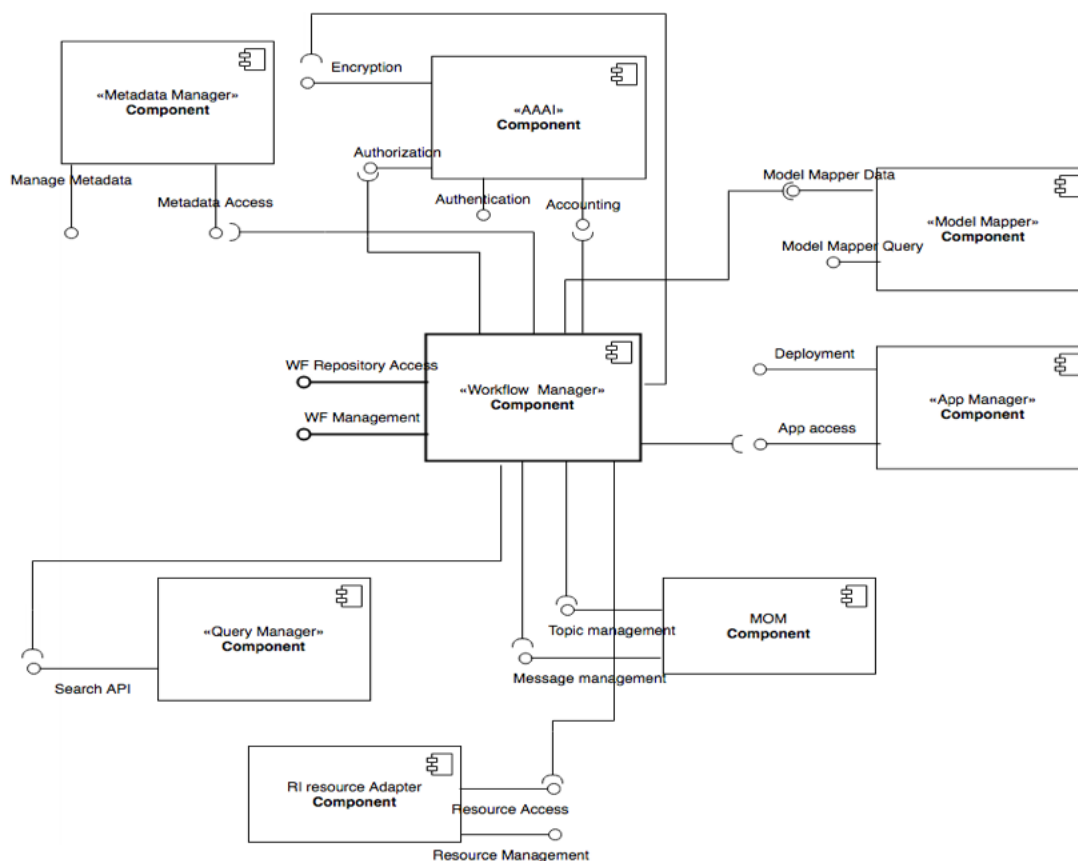


Figure 9 The Workflow Manager UML Component diagram

In the technical architecture, the functionalities of the Workflow Manager are implemented by the *Workflow Service* [D 33]. These functionalities have been partitioned in three groups, each one implemented by a different sub-component:

- The Workflow Configurator provides functionalities to build/edit/store workflows, and to control and monitor their executions
- The Workflow Executor manages the execution of workflows, including data staging, it may interact with the App Manager to execute tasks.
- The Workflow Repository provides functionalities to store and retrieve workflows descriptions to/from the Metadata Manager, and to publish them using the LD Manager.

Moreover we have added to the Workflow Service also the components of the Interoperability Manager: e-VRE WS, adapter and MOM.

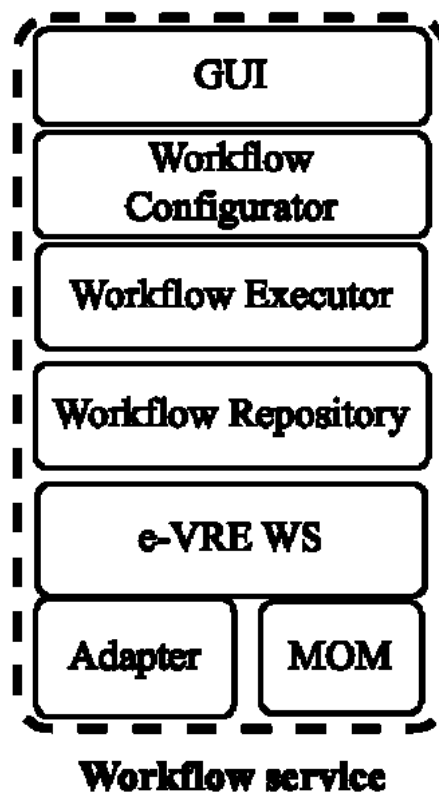


Figure 10 The Workflow Service

3.5.1 The Workflow Configurator

The main functionality of the Workflow Configurator is to access the VRE4EIC catalogue to:

- search for resources that may be used in building a Workflow, for instance Web services descriptions or references to external datasets, etc.
- save descriptions of web services, so these become e-VRE resources and can be searched and reused\

The UML sequence diagram in Figure 10, describes the interactions between e-VRE components in a use case where the Workflow Configurator search for WADL descriptions of Web Services.

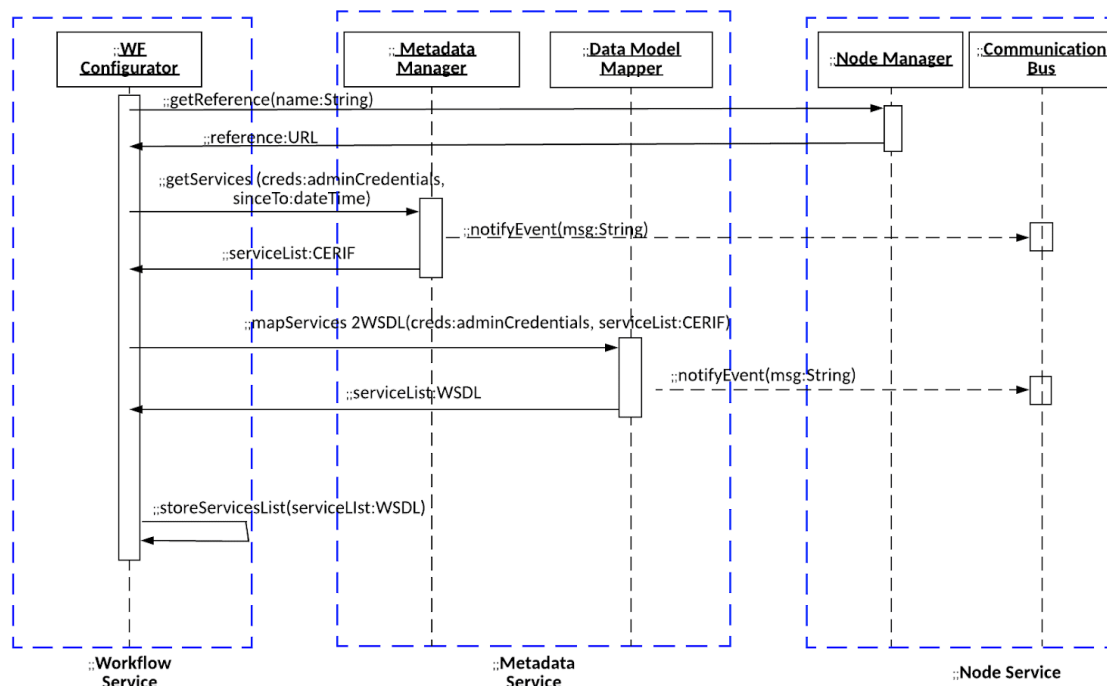


Figure 11 Search the catalogue for Web Services descriptions

The Workflow Configurator interacts directly with the Metadata Manager (via the Metadata Service e-VRE WS, not reported in the diagram), and to do this it needs to know the reference (the URL) of the Metadata Service. As explained in the section describing the Node Service, the Workflow Service obtains the reference of the Metadata Service by querying the Node Manager during its start-up phase. However, the reference to Metadata Service can be obtained also at runtime, this means that the Node Manager may implement a policy for load balancing when more Metadata Services are part of an e-VRE system.

3.5.2 The GUI, the Workflow Executor, the Workflow Repository

The Workflow Executor and the Workflow Repository components of the Workflow Service must implement a number of user requirements, as described in Deliverable 3.1, in particular the requirements: *PRQ28 Data Processing Control*, *ORQ2 Processing Parallelization*, and *ORQ4 Data Compartmentalization* [D 31]. The GUI should provide to users the possibilities of create workflows, save them in the Workflow Repository and execute workflows in the Workflow Executor.

The effort to obtain a complete implementation of these components exceeds the resources available in the project. For this reason, to implement the required functionalities we decided to investigate the possibility to integrate an existing workflow engine in e-VRE.

This starting point of our investigation has been a detailed analysis [HOLL] on state of art in the field of workflow engines. The following table summarizes results of the work:

	Kepler	Triana	Wings/ Pegasus	Taverna	Pipeline Pilot	KNIME
Life Science application support	++	++	+	+++	++	+++
GUI	+++	+++	+++	+++	+++	+++
Heterogeneous resources	+++	++	++	+++	++	+
Provenance & Sharing	++	++	++	+++	++	+
Scalability	+++	++	+++	++	++	++
Extensible	+++	++	++	+++	-	+
Security	++	+	+	++	+	+
Active development	+++	++	+++	+++	+++	+++

Table 3.1: Evaluation of common scientific workflow management systems. Legend: '-' means not supported, '+++' means fully supported.

A number of tests has been made on the above frameworks and two possible choices have been considered: Kepler and Apache Taverna. The main features of these two frameworks are compared in the table below.

Apache Taverna	Kepler
Domain-independent	Domain-independent
GUI for workflow composition	GUI for workflow composition
Simple Conceptual Unified Flow Language (SCUFL2)	Uniform access to computational components through actor model.
Enable to share and manipulate workflows outside the editor.	Workflows are saved as XML files
Dataflow-oriented model of execution and support loops	Many different models of computation are possible, focus on actor-oriented

Support for RESTful web services & OGC service consumption	
Support for the use of Cloud in workflow execution	Support for the use of Cloud in workflow execution
https://taverna.incubator.apache.org	https://kepler-project.org/
LGPL license	BSD License

We decided to use Apache Taverna for the prototype implementation. From the implementation point of view the main reasons for this choice have been that it enables third part developers to implement the web services and local services integration using Java related technologies; additionally it provides monitoring tools that can be easily integrated in e-VRE GUI, in particular the Taverna Workbench. The Taverna Workbench, a tool that enable users to create, manage and store workflows, has been designed and developed as a plug-in platform, this means that its functionalities can be easily extended by developing and installing new plug-ins. A specific plugin for e-VRE has been developed, it enables Taverna Workbench to interact with e-VRE. Details of the e-VRE plug-in are described in the following sections.

3.5.3 Workflow Service Implementation description

This section describes the implementation of the Workflow Service in terms of the three main actions of the *Fun15: Workflow Enactment* general function that is implemented by the Workflow Service [D 31].

3.5.3.1 Workflow creation

The workflow is created using the Taverna Workbench which access the VRE4EIC catalogue to get Web Services descriptions and use these descriptions to create workflows. The main responsible for this activity in e-VRE is the Workflow Configurator: it interacts with Metadata Manager to get the Web Services descriptions with AAAI to check authentication/authorization, with the Data Model Mapper to implement the transformation of CERIF records into a format accepted by Taverna.

To enable the Taverna Workbench to interact with the e-VRE we have developed a plug in module that uses the e-VRE WS provided by the Workflow Configurator. The plug-in is published and can be installed by any user using Taverna Workbench (release 2.5).

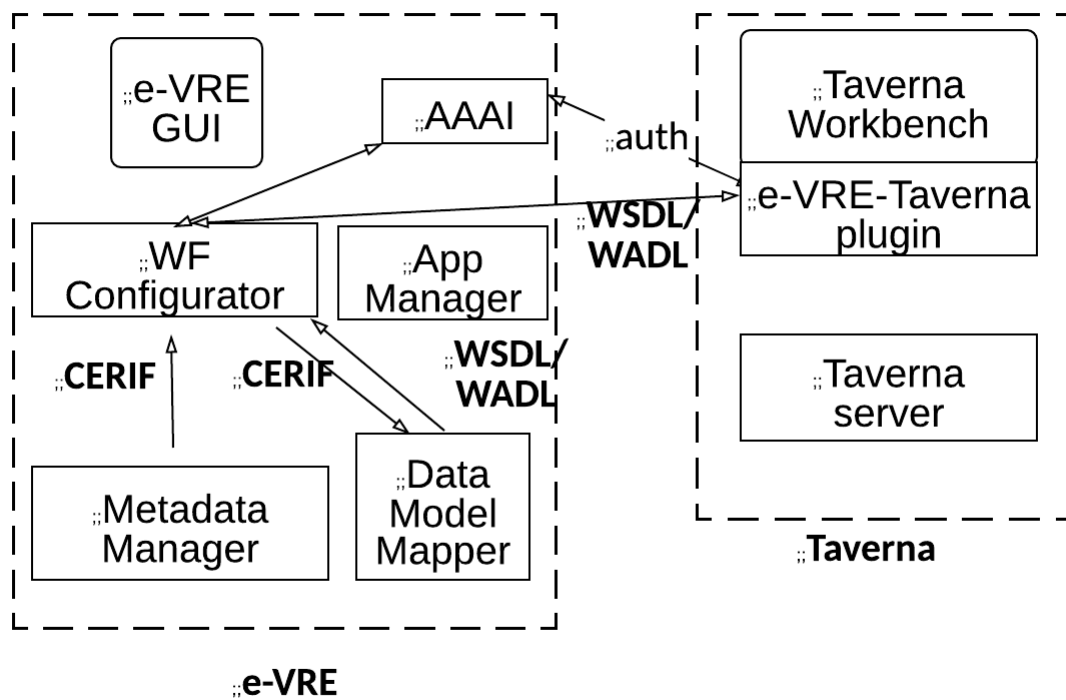


Figure 12 Workflow Creation using e-VRE-Taverna plugin

3.5.3.2 Workflow storage

When a workflow is created:

- It is stored in the local Apache Taverna repository,
- A WSDL document describing the workflow is created by e-VRE-Taverna plugin.
- The WSDL document is consumed by the Workflow Configurator of e-VRE, and passed onto the Data Model Mapper, which transforms it into a CERIFService description. The so obtained description is stored in the Metadata Manager (see Figure below).

From this point on, the workflow can be discovered and invoked as any other service whose description is stored in the e-VRE Catalogue.

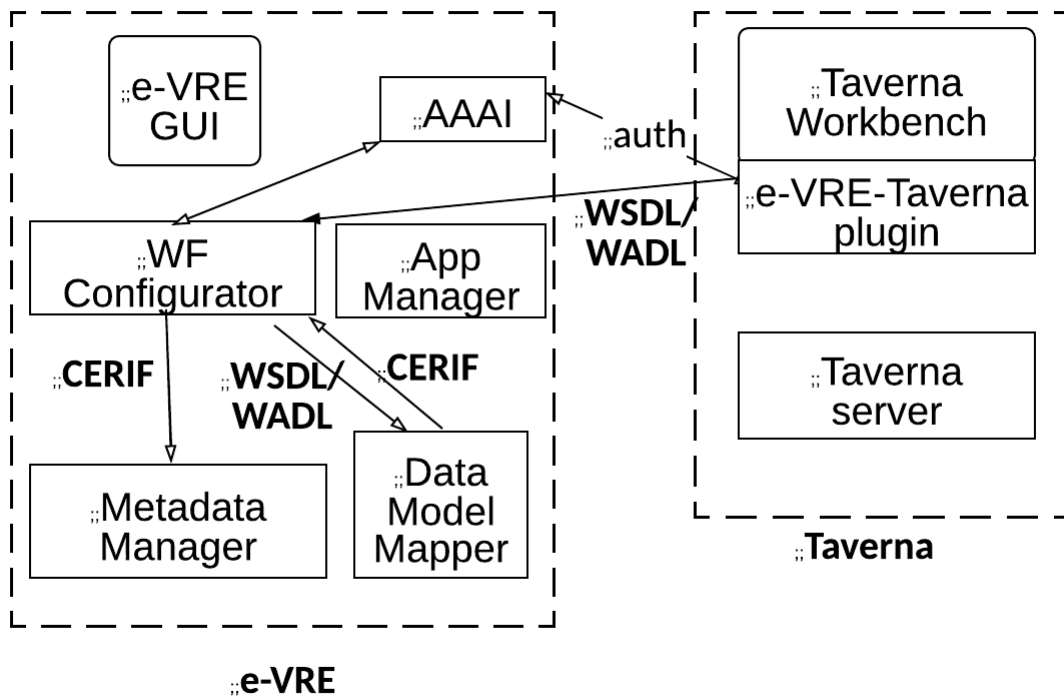


Figure 13 Storing a workflow description in the e-VRE Catalogue

3.5.3.3 Workflow execution

The execution of a number of predefined workflows can be launched by the user from the e-VRE GUI. The GUI captures the input parameters of the workflows (via an HTML form, or similar) and demands the execution to the App Manager of e-VRE. The App Manager, in turn, will interact with a defined Taverna server to execute the workflow. The result of the workflow is returned by the Taverna server to the App Manager, and from the App Manager to the GUI.

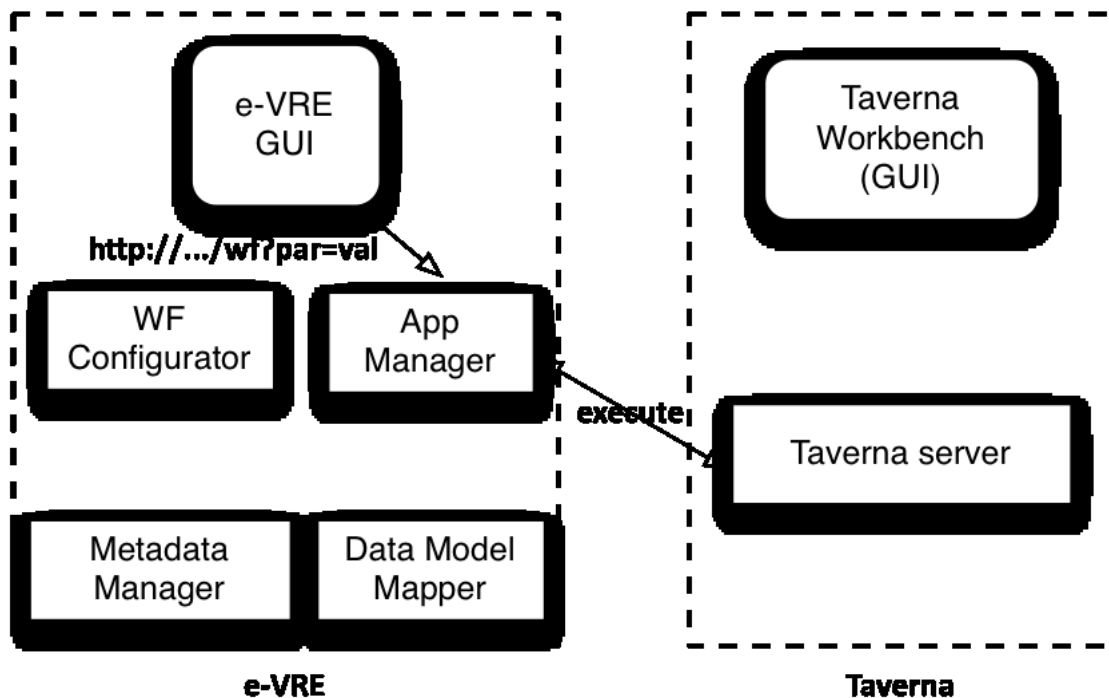


Figure 14 Executing a Workflow

3.5.4 Source code, documentation and set up

The java code of the Workflow Service is published on GitHub, and can be downloaded at the following URL:

<https://GitHub.com/vre4eic/WorkflowService>

The Workflow Service has been developed as a Java Maven project, at the moment, it only contains the source code of the Workflow Configurator.

The documentation of e-VRE Web Services is here:

<http://v4e-lab.isti.cnr.it:8080/WorkflowService/swagger-ui.html#/>

The eVRE-Taverna plugin source code is available here:

<https://GitHub.com/vre4eic/eVRETaverna>

The Readme.md file on GitHub repository contains instructions for the set-up of this building block and of the e-VRE Taverna plugin.

3.6 The e-VRE App Service

The goal of this building block is to implement the functionalities of the App Manager conceptual component: enable external applications to be embedded and used into the e-VRE system (see Deliverable 3.1 [D 31]). The App Manager is a crucial component for e-VRE, it should be implemented as a lightweight, unobtrusive software module that interacts with external applications to track their lifecycle and their usage.

Creating a generic App Manager, able to automatically manage lifecycle and usage for every possible external application embedded in the e-VRE, is not feasible: a plan has been defined in the design phase to consider the main standards for this purpose and to proceed implementing the App Service for these standards: the Servlet level 3 specification has been adopted as first candidate.

However, the e-VRE canonical prototype embeds two external software frameworks with a different integration technologies: the Telegram framework for notifications and Two Factor Authentication (2FA) using the Telegram API and the Taverna Workflow Engine, for workflow management GUI and execution using the plug-in platform provided by Taverna Workbench.

As described in the correspondent sections we have created two specific components each one running inside the e-VRE system and interacting with e-VRE building blocks using the Communication Bus.

The main role of the App Manager in e-VRE prototype is to consume messages sent by these components to keep a log of states transitions and to produce messages that are consumed by these two components, each one reacting according to its business logic.

4 The VRE4EIC Graphical User Interface

This section describes the implementation of a GUI for the VRE4EIC Reference Architecture. In particular this GUI facilitates the exploration, discovery and management of the metadata describing resources contained in the VRE4EIC catalogue. It incorporates a multitude of features on top of an intuitive and user friendly environment, in order for both novice and expert users to execute complex queries. The platform is agnostic to the underlying conceptual model, yet it can be configured to take advantage of the main concepts designed.

4.1 Introduction to this section

The publishing of structured and semantically enriched data is changing traditional models of conducting business and research. Modern science in particular is becoming more collaborative and multidisciplinary, taking advantage of the plethora of data being produced by groups with diverse scientific backgrounds. So called Virtual Research Environments (VREs) aim to promote this scheme, overcoming physical or semantic barriers, and facilitating researchers from diverse fields to exchange data and resources, decoupling science from ICT.

For such an interoperability to be achieved, and considering the heterogeneity in scope, features and technologies, various challenges are faced from the technical, semantic and legal standpoint. One major goal is the generation of high level ontologies with rich metadata that are easily explored by researchers.

4.2 Targeted Objectives of the Design

This section describes the design, architecture and implementation of the VRE4EIC Metadata Portal, a fully functional platform that facilitates the exploration, discovery and management of semantic metadata for both novice and expert users who wish to execute complex queries. The platform, provides an intuitive, user friendly environment that is highly configurable, making it appropriate for various domains, still being agnostic of the underlying conceptual model.

The purpose of the VRE4EIC Metadata Portal is to provide a user friendly environment to all VRE4EIC users for the search, management and import of Metadata, contained in certain VREs and RIs. This is achieved through the appropriate Graphical User Interface (GUI) through which end users can take advantage of the Node and Metadata Services.

The portal does not only aim at offering (another) simple query interface for providing access to the underlying metadata, hiding the complexity of writing expressive SPARQL queries. Although the look'n'feel resembles similar query building systems with the goal of reducing the learning curve for the novice user, the features incorporated are based on a thorough analysis of the requirements of existing VREs and Research Infrastructures (RIs), offering a repertoire of solutions for various beneficiaries. Overall, the goal is to support both

- Query writing for the expert, exploiting the capacity of the underlying conceptual models and gathering best practices from similar systems,
- Discovery of metadata (exploratory search) for the novice user, to help researchers search across domains and data that they are not familiar with, guiding them in the process of creating a query.

4.3 Functional Model

The current implementation offers a metadata catalogue containing metadata from affiliated RIs. In short, the main key features of the platform are described in the following paragraphs.

One key characteristic of the portal is that it dynamically executes sub-queries and presents partial results on-the-fly, while the user builds the query. The goal is twofold. On the one hand, it helps exploration by presenting results and allowing end users to limit the scope of the query and, on the other hand, it prevents from executing meaningless queries that return no results. This is achieved by transparently altering the options available during the query building process, eliminating choices that can lead to an empty search space.

The platform offers a combination of different ways for searching relevant metadata within the same query. It enables the user to compose queries using *keyword search*, *entity-based search*, *time range-queries*, *filter-based search* and *geo-spatial search* through an interactive map.

Filter-based search occurs based on an intuitive interaction model and may involve conjunctive and disjunctive nested queries. Options to limit the depth, the degree and the regular expression usage (AND/OR), are available, simplifying the excessive use of nested filters.

The screenshot displays the VRE4EIC Metadata Portal interface. At the top, the logo 'VRE4EIC Metadata Portal' is visible on the left, and a home icon with the name 'VANGELIS KRITSOTAKIS' is on the right. Below the header is a blue navigation bar with a search icon and the text 'Metadata Search', along with icons for share, star, settings, and help.

The main search area contains several components:

- Searching for...:** A panel on the left with a dropdown menu set to 'Person' and a text input field for 'Containing keyword'.
- Filter on entity "OrganisationUnit":** A blue arrow icon pointing to the right.
- Query Builder:** Three panels stacked vertically, each with a blue header and a close button.
 - Top Panel:** '... related' (relation: 'is member of') and '... to entity' (entity: 'OrganisationUnit'). It includes 'From Date' and 'Until Date' dropdowns, a 'Search by keyword' field, and a green 'Search By Keyword' button. A green '+' icon is on the right.
 - Middle Panel:** '... related' (relation: 'is coordinator of') and '... to entity' (entity: 'Project'). It includes 'From Date' and 'Until Date' dropdowns, a 'Search by keyword' field, and a green 'Search By Keyword' button. A blue '1' icon is on the right.
 - Bottom Panel:** Similar structure to the middle panel, with a grey 'OR' circle to its left and a blue '1' icon on the right.
- Region Set:** A panel on the right with a blue header and a close button, containing the text 'Marked Region containing 73 pins has been set'. A green '+' icon is on the right.
- Bottom Bar:** A grey 'SEARCH' button on the left and a pink 'RESET' button on the right.

Figure 15 Applying one filter on the target entity and another one on the related entity

Geo-spatial queries are constructed with the help of an interactive map, offering most of the functionalities that one expects to find in similar systems, such as searching by toponyms or geographical regions or selecting specific instances to be included in the query etc.

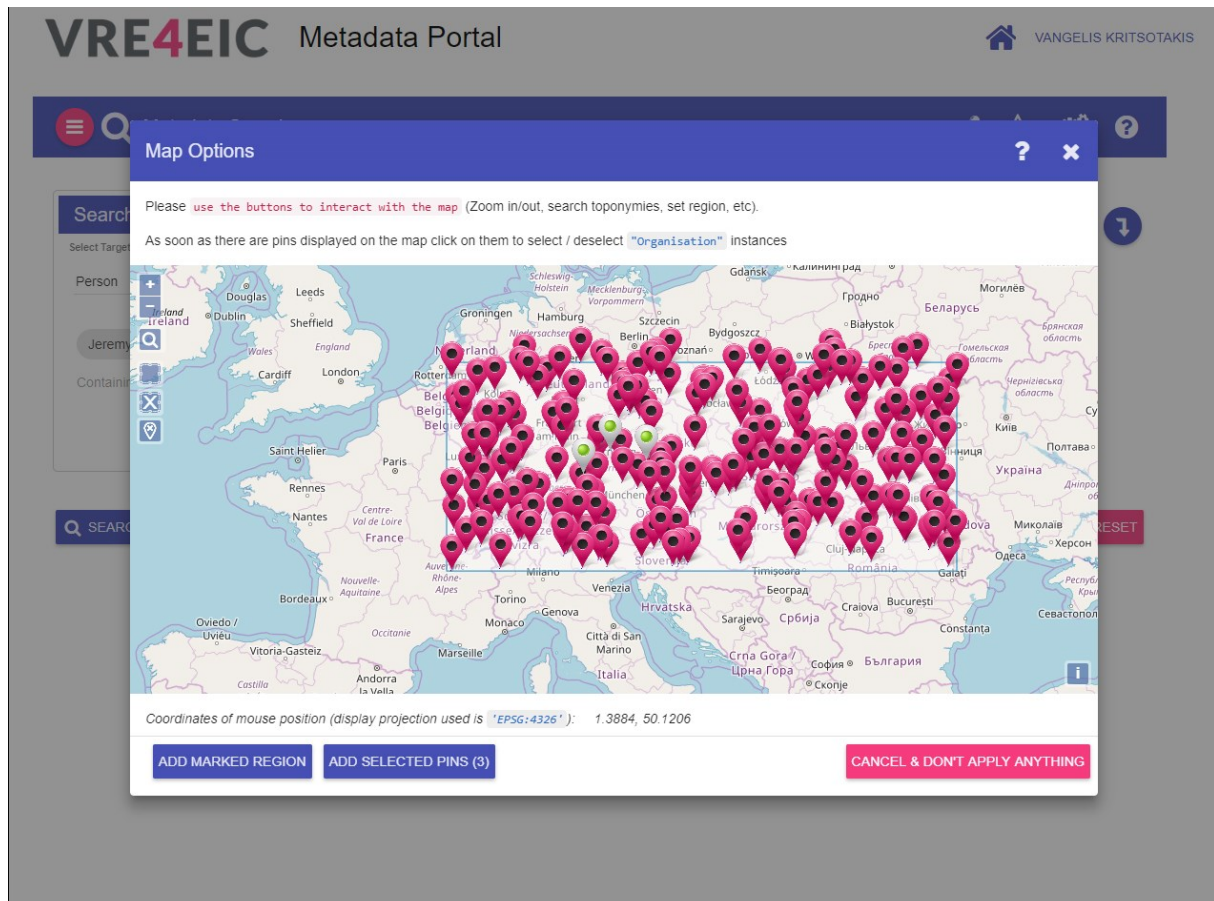
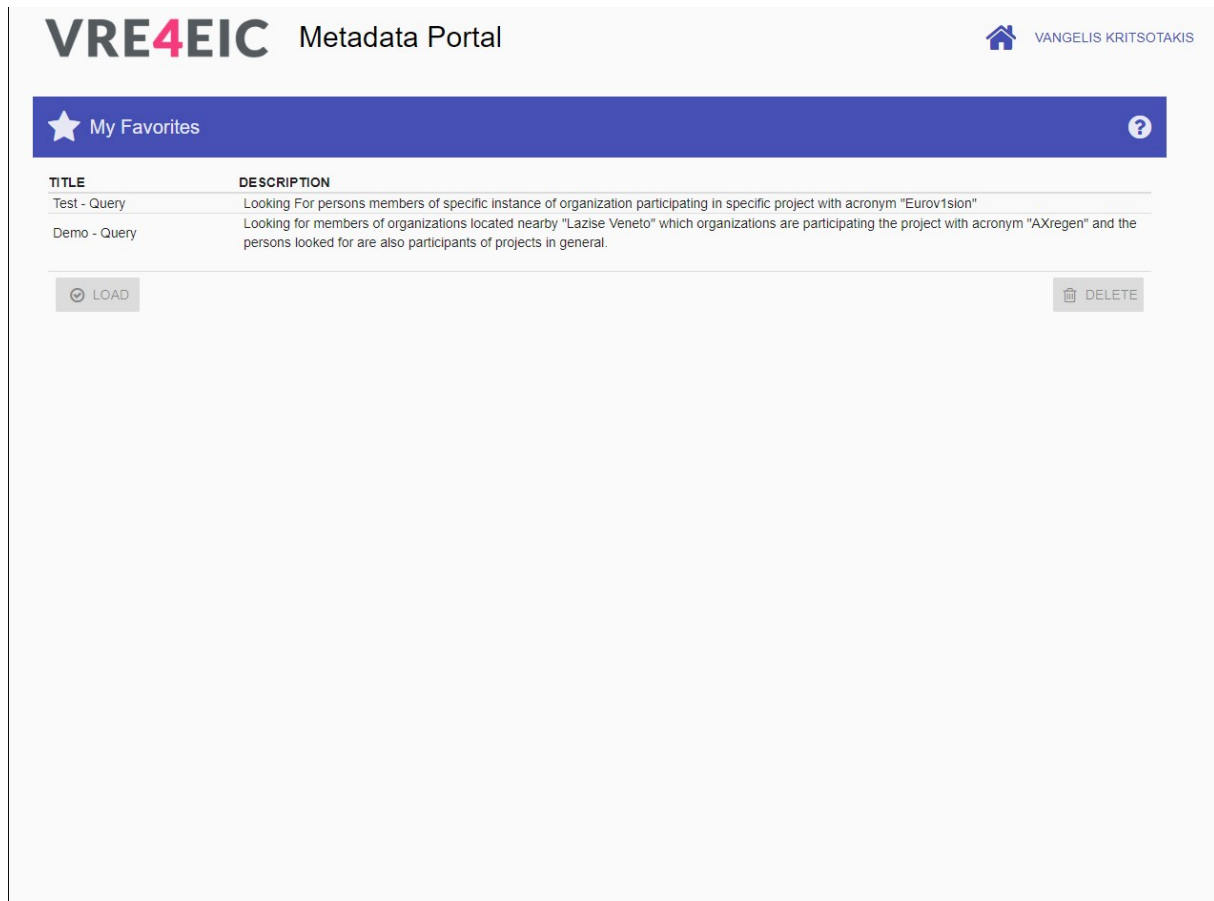


Figure 16 Setting geographical region and selecting instances by using the interactive map

The platform also gives users the ability to store queries for later use. This is a handy option, enabling the retrieval of complex queries that can be used as templates for constructing competency queries with multiple filters or for making minor adaptations to complex exploration tasks.



The screenshot displays the 'My Favorites' section of the VRE4EIC Metadata Portal. The header includes the VRE4EIC logo and the text 'Metadata Portal'. A user profile icon for 'VANGELIS KRITSOTAKIS' is visible in the top right. The main content area features a blue header with a star icon and the text 'My Favorites'. Below this is a table with two columns: 'TITLE' and 'DESCRIPTION'. The table contains two entries: 'Test - Query' and 'Demo - Query'. At the bottom of the table, there are two buttons: 'LOAD' and 'DELETE'.

TITLE	DESCRIPTION
Test - Query	Looking For persons members of specific instance of organization participating in specific project with acronym "Eurov1sion"
Demo - Query	Looking for members of organizations located nearby "Lazise Veneto" which organizations are participating the project with acronym "AXregen" and the persons looked for are also participants of projects in general.

Figure 17 List of queries stored into user's favorites

After executing a query, the results can be examined through an internal resolver, allowing simple navigation from any instance of the targeted entity to any of the instances of the related entities. In that way, end users can even continue their discovery after the results are retrieved.

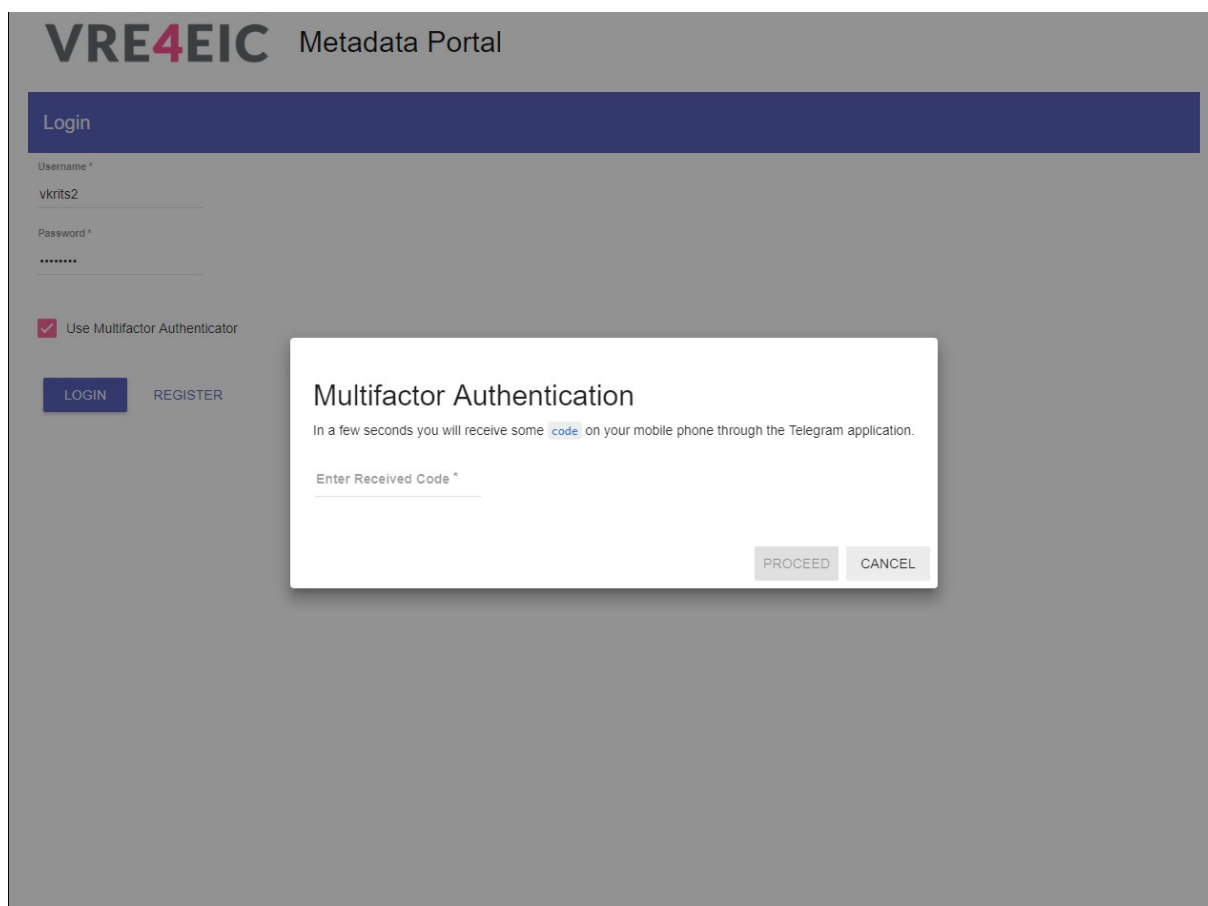
The screenshot displays the VRE4EIC Metadata Portal. The main interface is divided into several sections:

- Header:** "VRE4EIC Metadata Portal" and "Information regarding the selected OrganisationUnit".
- Search Bar:** "Metadata Search" with a search icon and a menu icon.
- Search Filters:**
 - Searching for...:** "Select Target Entity" dropdown set to "Person", a search input containing "Jeremy", and a "SEARCH" button.
 - ... related:** "Select Relation" dropdown set to "is member of", with "From Date" and "Until Date" filters.
- Data Results:** A list of names under the heading "Data Results". A note above the list says: "To resolve any of the 'Person' results listed below, please click on the respective". The list includes:

NAME
Jeremy Stuteville
Jeremy Mizukami
Jeremy Clingenpeel
Jeremy Brabec
Jeremy Mlynek
Jeremy Barbini
Jeremy Taira
Jeremy Bourbois
Jeremy Crossno
Jeremy Mckissack
- Entity Detail View (Right Panel):**
 - Entity: "DEUTSCHE NATIONALBIBLIOTHEK"
 - Type: "Person" (highlighted in pink)
 - Relationships:
 - has member: Jeremy Stuteville
 - has member: King Dante
 - has member: Delinda Yorty
 - Type: "Project" (highlighted in pink)
 - Relationships:
 - is participant in: Keeping Emulation Environments Portable KEEP
 - is participant in: Alliance Permanent Access to the Records of Science in Europe Network APARSEN
 - is participant in: Implementation of quality indicators in Palliative Care sTudy IMPACT
 - is participant in: (partially visible)

Figure 18 Navigating through results

The platform provides a multi-factor authentication mechanism for granting access to the users. This mechanism requires users to present two pieces of evidence, which are their regular credentials and a code sent to the "Telegram Messenger" account, they possess. This compartment also provides Role Based Access Control (RBAC), ensuring that users actions are regulated according their knowledge background determined through their user roles.



The screenshot displays the VRE4EIC Metadata Portal login interface. At the top left, the logo 'VRE4EIC' is followed by 'Metadata Portal'. Below this is a dark blue header with the word 'Login'. The main form area contains a 'Username *' field with the value 'vkrits2', a 'Password *' field with masked characters, and a checked checkbox for 'Use Multifactor Authenticator'. There are 'LOGIN' and 'REGISTER' buttons. A white modal window titled 'Multifactor Authentication' is centered on the screen. It contains the text: 'In a few seconds you will receive some `code` on your mobile phone through the Telegram application.' Below this is an 'Enter Received Code *' input field. At the bottom right of the modal are 'PROCEED' and 'CANCEL' buttons.

Figure 19 Login using two-factor authentication

The platform allows data import from a variety of RDF file formats on either existing or new defined graphs. The process is as simple as a drag & drop and supports multiple file upload in a single step. During this process, the system acquires any available user-profiling material and uses it as extra provenance information to accompany the imported data. This provenance information is important for knowing who has import what and where.

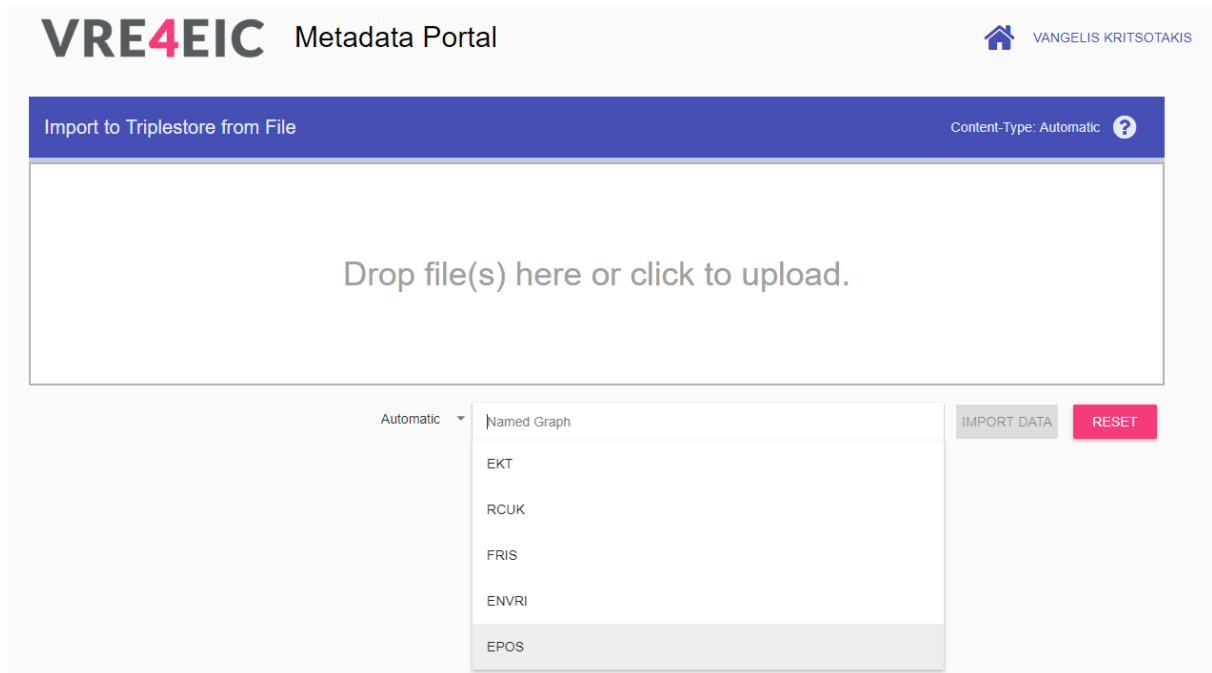


Figure 20 Importing RDF data by dragging and dropping files

Finally the platform is configurable on many different levels, based on a specialized dashboard, in order to enhance flexibility, robustness and simplicity. Configuration options are accessed directly through the GUI (an administrator user role is required) and can significantly affect functionality, system performance and the level of complexity. These parameters are mainly limitations, regulations and option exceptions to be set on queries, the logical expressions used, or the available entities.

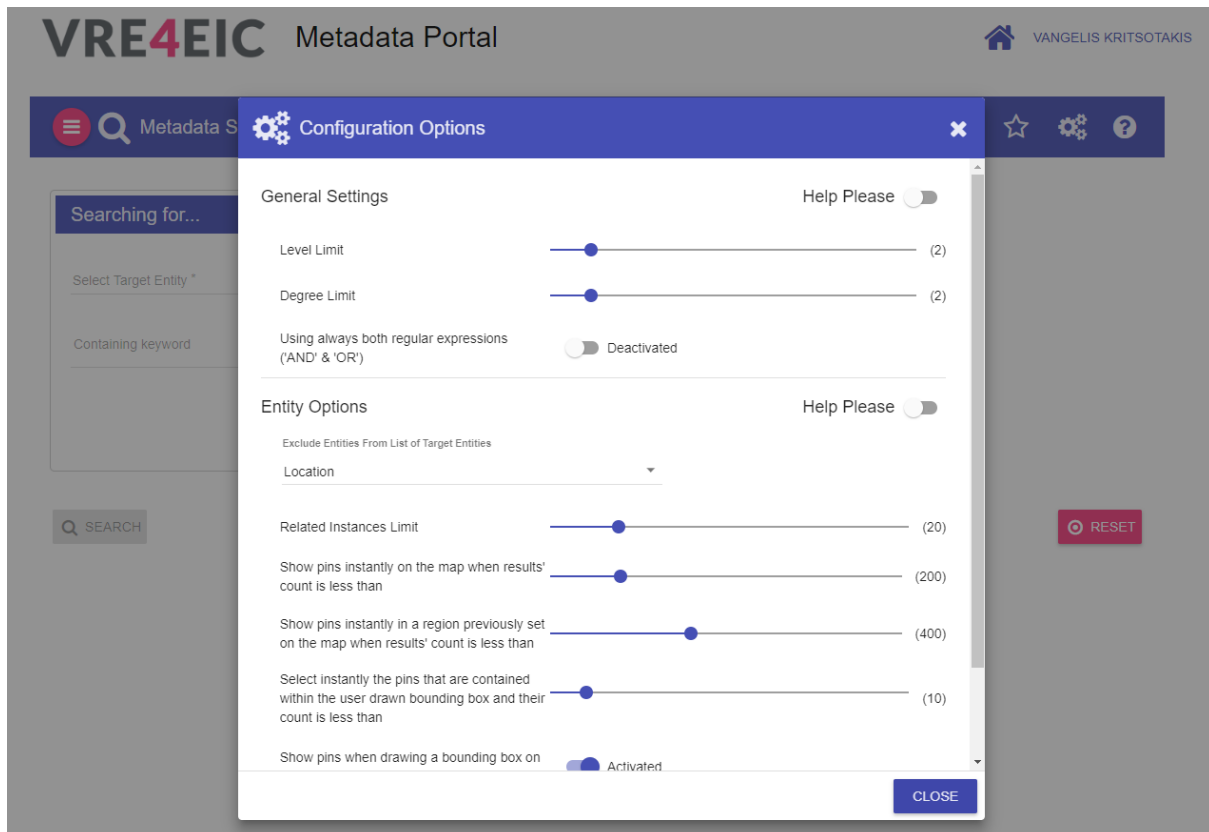


Figure 21 Configuration options that can enhance performance and set the desired usage complexity

Overall, a more detailed listing of functionalities supported by the platform is provided in Annex section of this deliverable. The functionality covers aspects related to security, data presentation and discovery, data import and export, system configuration and administration and system's robustness and fault tolerance.

4.4 Architecture of the VRE4EIC GUI

The GUI is implemented by several sub-components which interact with external components (mainly using restful web services). A high-level diagram of the platform is presented in Figure 22.

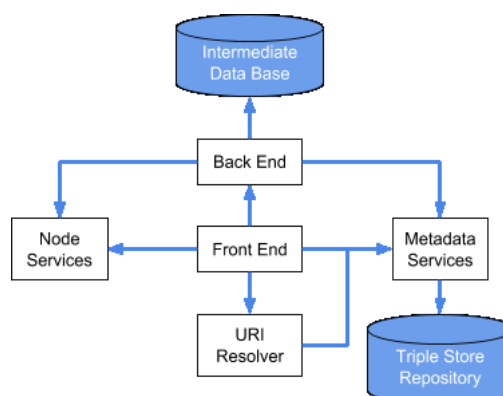


Figure 22 VRE4EIC Architectural Design w.r.t. sub-components

In the next sections, each of the components, constituting the portal, are described in more details along with their responsibilities and main functionality.

4.4.1 Front End

The front end is mainly responsible for facilitating human-computer interaction and provides the required features that constitute the GUI intuitive and usable. However, its functionality is further expanded, since it is also responsible for implementing the required logic for executing users' actions and deciding the proper services to be called. Moreover, this component is responsible for properly handling errors and informing end-users about them when they occur. Finally, tasks related with users' login or registration are directly handled by this component. The front end is based on the Model View Controller (MVC)¹⁰ design (Figure 23), where the user interface layer is isolated from the application's logic. The available controllers receive http requests and dynamically build the required models for the data view. This view then uses the data prepared by the controller to generate a final presentable response.

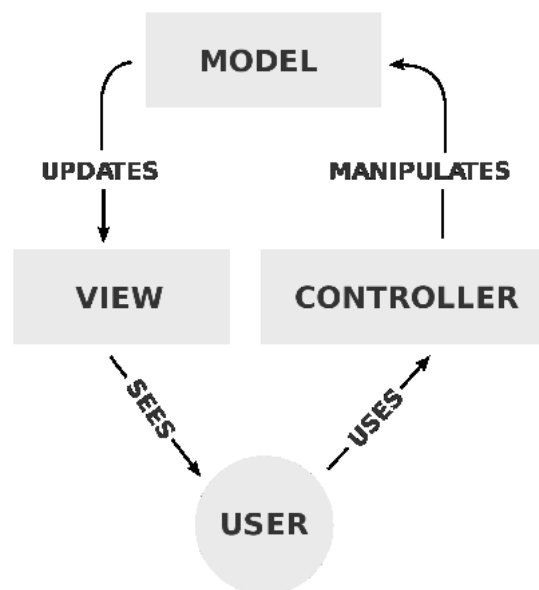


Figure 23 Diagram of interactions within the MVC pattern

The logic of the front-end regarding the dynamic construction of queries, relies on the general assumption that the user is looking for a “target entity” which is “related” to one or more “entities”. That assumption is further expanded so that each of the “entities” can be “related” to other “entities”, forming in this way a tree model on the fly, describing the constructed query. The picture below shows the tree model in a generic way.

¹⁰ <https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

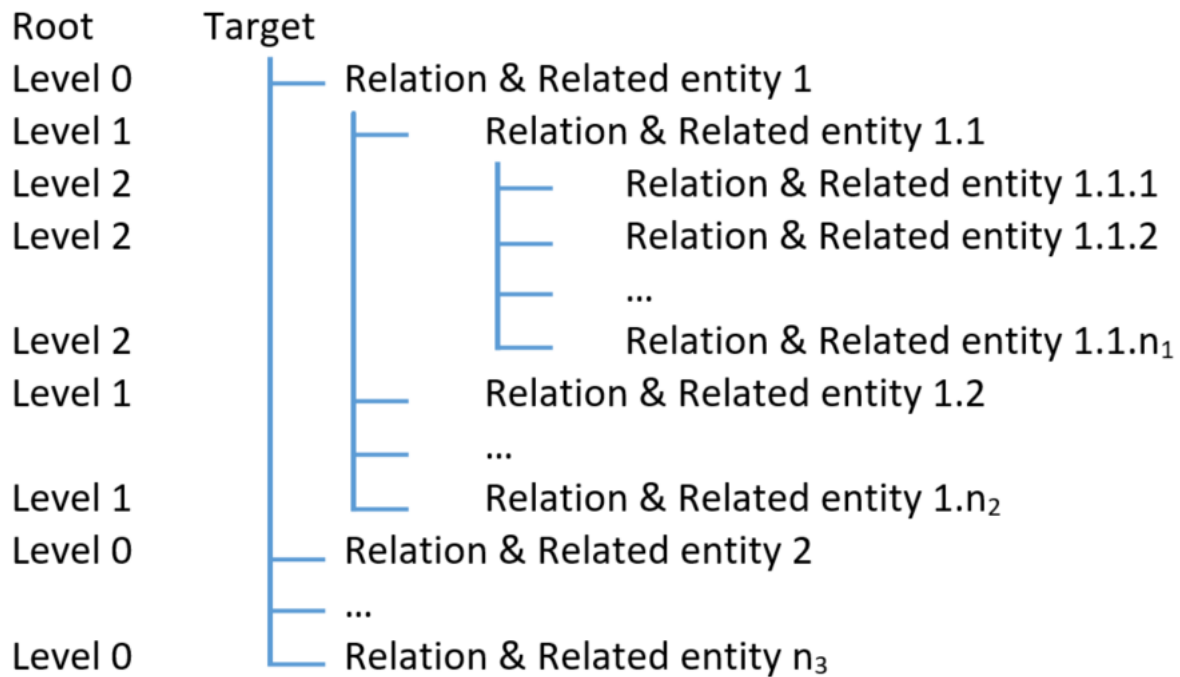


Figure 24 Tree model describing the general concept under which relies the logic of dynamically constructing queries

All nodes under the root node are actual filters applied on target entity. The same applies all node under any node parent. Any nodes in the same level that have the same parent and are more than one are accompanied by a regular expression (OR/AND) that defines the way filters are applied together.

4.4.2 Back End

The back end is responsible for serving all front end needs by executing the required functions and calling the appropriate web services. The back end mainly consists of controllers, internal back-end services (not to be confused with web services) and an Intermediate database.

Each controller is responsible for calling the appropriate services or external web-services, which combined together can fulfill a task. Moreover, the responsibility of the controllers is to deliver the required output at the front end, or the appropriate error message, if some failure occurs. This sub-component is responsible for checking the token's validity in any interaction with the front-end and acts respectively. On the other hand, services are responsible for accomplishing more specific tasks that usually aim on single targets. These services interact with web services to achieve their goal. The involved web services are the node services, related with user's profile and security, and the metadata services, related to query execution and metadata import / export, which are explained next. Since the platform interacts with users and is flexible enough to operate in a generic form, configuration options and users' structured queries, have to be stored to some place different than the Triple Store Repository itself, since this is only used for storing metadata. As such, an intermediate database is used for serving this purpose.

4.5 Interactions of the VRE4EIC GUI with the e-VRE building blocks

4.5.1 Node Service

The VRE4EIC Metadata Portal interacts with the Node Service for:

- User authentication at login;
- User registration;
- Retrieving user's profile information;
- Retrieving User roles for applying RBAC;

It is important to mention that the portal itself does not hold any personal information related to end-users, since this is the responsibility of the Node Services to fulfill in some remote and secure repository.

4.5.2 Metadata Service

The Metadata Service is the building block, responsible for providing services related to metadata management. These services can be further classified into Query Services and Import/Export Services. Query services execute queries on the data stored into the Triple Store Repository and deliver the output back to the requester. Import services can insert data formed in a variety of formats into the Triple Store Repository. Finally, Export Services can extract data specified from the Triple Store repository and deliver it back to the requester in a variety of formats. The Metadata Restful API are documented using Swagger, documents can be accessed using this link:

<https://app.swaggerhub.com/apis/rousakis/ld-services/1.0.0>.

The set of specific Web Service entry points used in the VRE4EIC Portal appear in Figure 25.

GET	/query/virtuoso	Executes a SPARQL query
GET	/query/virtuoso/count	Converts a sparql query into a query which counts the query results
GET	/export/virtuoso	Exports the contents of an RDF dataset
POST	/import/virtuoso	Imports an RDF data string
POST	/update/virtuoso	Executes a SPARQL update statement

Figure 25 The list of entry points and descriptions of Metadata Service e-VRE WS used by VRE4EIC Portal

4.6 Technologies used in implementing the VRE4EIC GUI

A powerful combination of technologies has been used to implement the VRE4EIC Graphical User Interface

- Spring Boot (a Web Application Using Spring MVC)
- AngularJS (a structural framework for dynamic web apps based on HTML and JavaScript)

- Bootstrap-UI & Material Design (UI component frameworks)

All the components can be executed from the command line as standalone Maven applications, since they include an embedded server container (Jetty by default, however Tomcat can also be embedded)

- No prior installed software is required (except of the JVM);
- No prior configuration is needed;
- Independent, portable and easy to be deployed

5 The Enhanced EPOS VRE

EPOS, the European Plate Observing System, is a long-term plan to facilitate integrated use of data, data products, and facilities from distributed research infrastructures for solid Earth science in Europe. EPOS will bring together Earth scientists, national research infrastructures, ICT (Information & Communication Technology) experts, decision makers, and public to develop new concepts and tools for accurate, durable, and sustainable answers to societal questions concerning geo-hazards and those geodynamic phenomena (including geo-resources) relevant to the environment and human welfare.

EPOS vision is that the integration of the existing national and trans-national research infrastructures will increase access and use of the multidisciplinary data recorded by the solid Earth monitoring networks, acquired in laboratory experiments and/or produced by computational simulations. The establishment of EPOS will foster worldwide interoperability in the Earth sciences and services to a broad community of users.

Many institutional data providers in the field of solid Earth science exist in Europe, and they provide access to datasets and other resources at National level. However only some of them are federated and integrated, and as a result a user willing to make use of resources, (that is to say datasets, data products, software or services), has to access institutional portals, department databases and other scattered sources of information. That's where EPOS comes into action. Its main goal is to integrate scattered resources across Europe in the field of Solid Earth Sciences. That's why EPOS is also referred to as a "long-term plan for the integration of research infrastructures for solid Earth Science in Europe», with the goal of investigating tectonic processes, earthquakes related phenomena, volcanic eruptions, surface dynamics and geo resources.

Research Infrastructures and Facilities encompassed by EPOS are located in 25 countries, mostly European countries. They also include International organizations like ORFEUS and EMSC in seismology, INTERMAGNET in magnetic observation discipline, EuroGeoSurveys in geology and others. The amount of resources to be integrated is huge, thus making the challenge of building EPOS exciting: We want indeed to: Integrate more than 250 National and Regional Research Infrastructures; Provide access to data produced by around 10.000 sensors in Europe; Manage and give access to several petabytes of data. Thousands of users are expected, so EPOS also need a robust user management system.

In the framework of VRE4EIC, EPOS had a twofold synergy with the e-VRE developed in the project, at integration level and at enhancement level

5.1 EPOS Integration within VRE4EIC

In the framework of VRE4EIC EPOS contributed by providing resources, dataset, webservice and knowledge in order to make its assets available in an integrated way through VRE4EIC e-VRE prototype.

5.1.1 Provision of metadata describing EPOS assets

First step was to provide the “assets” of EPOS described by means of metadata that was ingested into the main VRE4EIC catalogue.

Currently EPOS provided more than 200 metadata elements, of which:

70 persons, 35 organisations, 96 webservices, 6 datasets.

5.1.2 Provision of Scientific Background, use case and tools

VRE4EIC prototype focuses on the provision of integrated resources by research infrastructure and VREs. However in order to demonstrate the real contribution of VRE4EIC e-VRE, such resources needs to be provided, consumed and organised in a meaningful way, so that researches can eventually obtain information and data of interest.

With respect to the quality and type of information provided to VRE4EIC, also with the goal of building a scientifically meaningful demonstrator, EPOS provided the scientific background, based on literature (L. Chiaraluce, Unravelling the complexity of Apenninic extensional fault systems: A review of the 2009 L'Aquila earthquake (Central Apennines, Italy), Journal of Structural Geology, Volume 42, 2012, Pages 2-18, ISSN 0191-8141,

<https://doi.org/10.1016/j.jsg.2012.06.007> and other studies).

The natural consequence was also the provision of user story, based on the scientific background, and of meaningful datasets and software tools.

User story

Here is the user story provided by EPOS:

“One of the added values of Virtual Research Environments, and in particular of VRE4EIC, is the capability of aggregating resources by several data providers. It is clearly a requirement in many domains. For instance in the Solid Earth Sciences domain, tectonic studies take advantage of all information provided by different disciplines and by different sensors type: seismometers, GPS systems, Satellites.

This is the case of the paper [1] L. Chiaraluce, “Unravelling the complexity of Apenninic extensional fault systems : A review of the 2009 L’Aquila earthquake (Central Apennines , Italy),” J. Struct. Geol., vol. 42, pp. 2–18, 2012.

That’s a study of the Apenninic fault systems which includes elastic properties of the fault, and modeling of the processes that took place during the event, in this case dilatancy and diffusion processes.

This study made use of:

- 1. Historical seismic sequences*
- 2. Seismic datasets, in particular seismic waveforms from seismic stations, and earthquake localisation*
- 3. Geological maps (Carbonates platform)*
- 4. Earthquake focal mechanisms*
- 5. InSAR imagery*
- 6. other datasets*

Some of the above resources are available by means of services set up by institutions or Research Infrastructure. This is the case, for instance, of historical earthquakes, seismic waveforms and localisation, some geological maps, InSAR images.

However those sources of data are available from different sources, which sometime require different user/password pairs, access to several portals, with different methods. Sometimes they are even difficult to search or access due to the lack of appropriate user interfaces.

In this context a VRE that can integrate different sources and provide access to resources by means of a simple search and discovery tool, would enable scientists to have a full overview and full access, from one single portal, to all (or almost all) needed resources.

On the basis of the described example, we will now use the VRE4EIC GUI to get access to some of the resources used in the paper we just mentioned.”

5.1.2.1 Datasets and Software

In order to demonstrate the real added value of a eVRE, on the basis of the above user story, EPOS provided datasets related to earthquakes and volcanos (seismological events, seismic waveforms, and volcano related measurements like data flows), webservice description to obtain seismic waveforms and earthquakes events details, and software to process and visualise seismic waveforms (SeisGram2K Seismogram Viewer).

5.1.3 EPOS integration within VRE4EIC - Workflows

In the framework of integrating EPOS resources in VRE4EIC, EPOS also provided access to so called “scientific workflows”. The following diagram (Figure 26) shows how the integration was done:

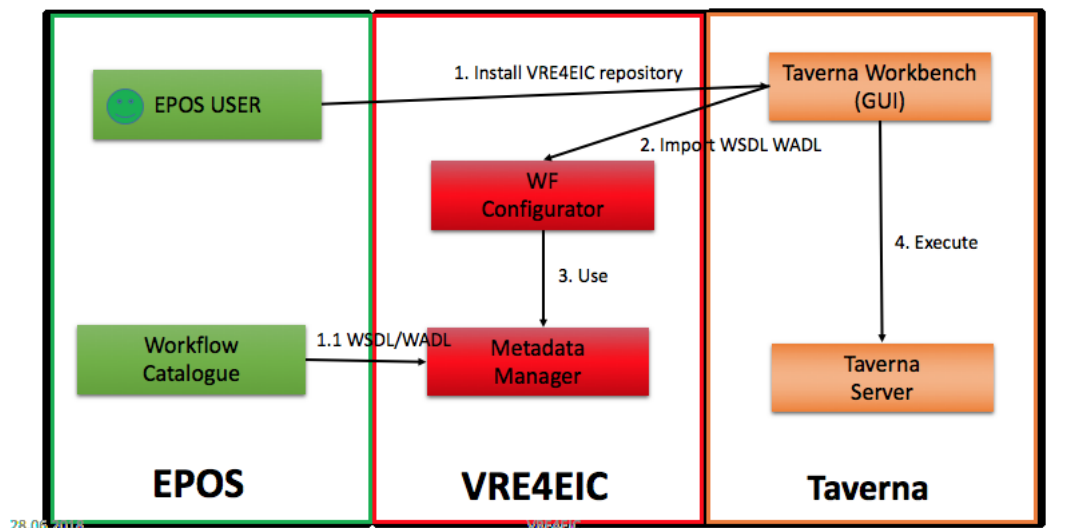


Figure 26 Epos e-VRE workflow integration

The three boxes represent respectively EPOS system, VRE4EIC system and the “user system” (e.g. laptop).

Initially an EPOS user, but might be any user, launch on his/her own laptop the TAVERNA workbench application in order to execute some scientific workflow (step 1 in the picture). In order to access to workflows provided by the VRE4EIC system, the user install a plugin that automatically connects to the WF configurator component (in the VRE4EIC domain) and fetches web services descriptions stored into VRE4EIC metadata manager (step 2 and step 3 in the picture). The metadata manager, in turn, access to webservice description from EPOS workflows catalogue (whether runtime or by ingesting information in advance).

This ensures that any non-skilled user can take advantage of workflows and webservice from EPOS domain (but potentially from any domain) just by installing a plugin on its workflow application (in this case Taverna workbench).

5.2 EPOS enhancement by means of VRE4EIC building blocks

In addition to the integration, EPOS took advantage of: (a) VRE4EIC know-how in terms of architecture paradigms and techniques, and (b) of VRE4EIC services, in particular the AAAI service.

5.2.1 EPOS architecture enhancement

First enhancement in EPOS was in terms of system architectural paradigms and functionalities. Leveraging on the studies, tests and experience from WP3, EPOS technical team started a fruitful collaboration. Both system start from a common baseline, which can be summarized with the a) adoption of the same architectural paradigm (Microservices), b) adoption of the same metadata oriented approach, through the use of a CERIF based metadata catalogue.

EPOS technical staff and VRE4EIC WP3 had regular meetings and discussion, both remotely and face to face (at the Project’s meetings) where the two architectures were compared.

As a result, EPOS architecture was enhanced by moving from a microservice centralized management, to a microservice choreography approach, both described in Section 3 of this Deliverable.

A snapshot of the EPOS Architecture follows.

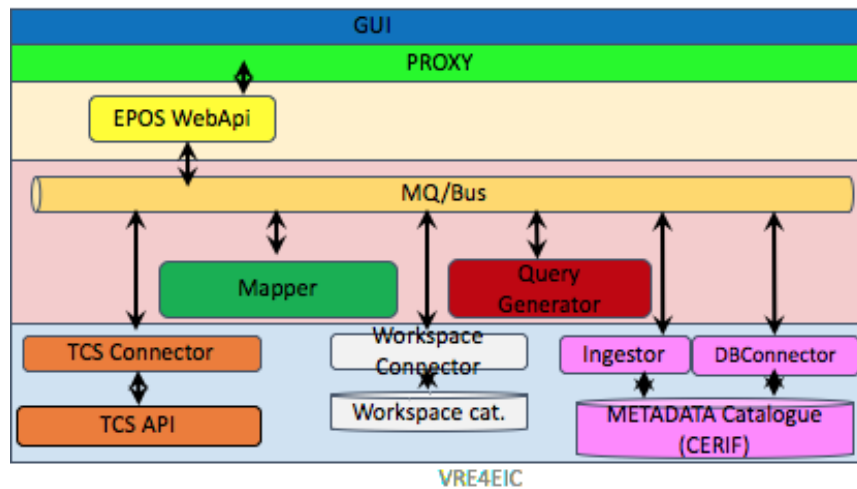


Figure 27 The EPOS architecture

5.2.2 EPOS functionality enhancement

A second enhancement of EPOS was in terms of additional functionalities provided by VRE4EIC.

As mentioned, the main EPOS system, called Integrated Core Services Central hub, uses a microservice approach. It includes a central queuing system and several components that performs atomic tasks and are connected to the queuing system. Examples of microservices are reported in the above in the diagram: the queueing system, the connector to Thematic Core Services and the metadata catalogue and others.

As evident from the diagram, some components implementing functionalities as authorisation are missing. They are however implemented by components or “building blocks” developed in the context of VRE4EIC. Building blocks are basically the components of the VRE4EIC reference architecture. Technically each building block is a microservice implementing a major component of the Reference Architecture, so it can be easily “plugged” into other architectures.

In this case, the building block to consider is the «AAAI service». AAAI stands for Authentication, Authorization, Accounting Infrastructure, that is to say a system to manage user secure access to a system with authorization. The main characteristic of this service is that it integrates different authentication mechanisms in one single system, thus providing a user a single point of access to resources. In practice, it means that independently from his or her account credentials, user will type them in one single form. Integrating it into an existing Research Infrastructure like EPOS, means avoiding the effort of integrating many different authentication services into one Research Infrastructure, with all related technical and security issues.

EPOS tested the module, and the results were encouraging, as it enabled EPOS users with existing credentials to log in to EPOS in an easy way, without jumping from a website to another. It indeed integrated several heterogeneous Identity Providers, like eduGAIN, Google, but potentially – what we aim at doing in the future – also other providers, both academic and generic, for instance ORCID, GitHub, Facebook.

The following diagram shows in terms of functional blocks how the enhancement was achieved.

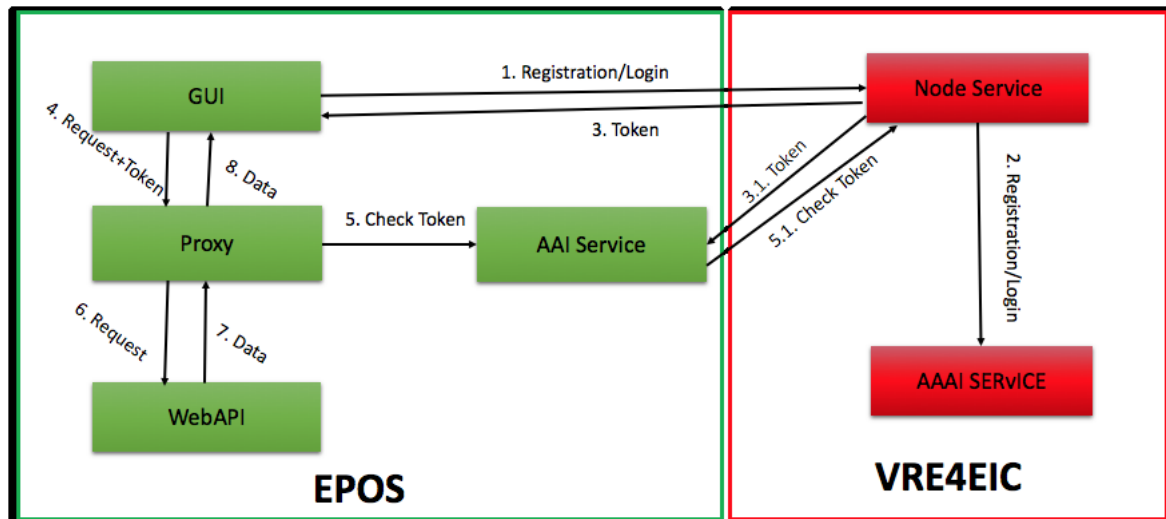


Figure 28 EPOS enhancement

Green box represent the EPOS system, while the red box represents the VRE4EIC system prototype.

Users access to the EPOS GUI, and when they want to perform login they are redirected (step 1) to the VRE4EIC node services. This service in turn register/login the user by means of the VRE4EIC AAI service (step 2). Response is enclosed into a token that can be used from the GUI (step 3). Such token will then be used by the GUI for communicating with the EPOS WEB APIs. Anytime the GUI makes a call the EPOS WEB APIs, indeed, the token goes through a proxy that checks its validity by connecting to the VRE4EIC service (steps 4-5). If the token is recognised as valid, then the request proceeds (step 6) and WEB APIs respond with a data payload that is then rendered or presented by the GUI to the user (steps 7-8).

These components interoperate in real time, thus providing the functionality of logging in to the EPOS Web Interface by using the services provided externally by VRE4EIC. All this complexity is hidden to the user, who just insert login and password in one simple login form.

A working demo of this integration can be found here:

<http://nodedev.bgs.ac.uk/epos/epos-gui/otherAAIversion/search>

Clicking on the “login” top right button, user can log in and is automatically recognised by the system.

5.3 Conclusion of this section

The collaboration and interaction between EPOS and VRE4EIC was carried along two main dimension: EPOS integration and EPOS enhancements.

In the first one, EPOS contributed to make its assets (metadata, datasets etc.) available in an integrated way through VRE4EIC system prototype. In the second one, EPOS took advantage of existing building blocks from VRE4EIC that implemented missing functionalities in EPOS. This interaction hence demonstrated both the feasibility of the VRE4EIC concept that aims at integrating heterogeneous resources in a homogenous way and the added value of the architecture and developments carried on in VRE4EIC that can contribute to the enhancements (i.e. expanding functionalities) of existing VREs and Research Infrastructures.

6 The Enhanced ENVRIplus VRE

The ENVRI community represents a cluster of environmental and earth science research infrastructures, and thus represents a vital forum in which to promote the e-VRE solutions developed in the project. The Data for Science theme within the ENVRIplus project is concerned with providing common technical solutions and recommendations to many of the problems shared by the ENVRI community, for example with regard to metadata cataloguing, provenance, identification and citation of persistent resources and data processing. Thus the e-VRE architecture and building blocks solutions have both been exploited to the ENVRIplus community as part of the ENVRI service portfolio of technologies, standards and recommendations. More specifically, e-VRE developments have been applied to the problems of enhancing i) cross-RI data and service discovery, ii) cross-RI workflow composition, and iii) cross infrastructure workflow execution and provenance.

6.1 Community catalogue for cross-RI data and services

In the ENVRIplus community, data and other digital assets are catalogued using different metadata standards (e.g., CKAN, ISO 19139 and Dublin Core) and using different technical solutions to serve metadata (e.g., B2FIND, GeoNetwork and Get-IT). Such diversity makes cross-RI data and service discovery very difficult.

To address this, the e-VRE solution has been used in ENVRIplus to build a contextual rich community catalogue for multiple research infrastructures in ENVRIplus. The CERIF standard was included in the ENVRIplus catalogue recommendation together with CKAN (used by EUDAT's B2FIND service). More specifically, the following two actions were taken to address the short term and long term challenges respectively:

6.1.1 Short term: manually setting up a CERIF-based data catalogue

In the short term, the CERIF database and catalogue software stack already implemented by EPOS (which is also a member of the ENVRI community) have been directly deployed in ENVRIplus by ENVRIplus project partner IFREMER. By setting the ENVRIplus instance, a small set of records from SeaDataNet (concerned with the marine domain), ICOS (concerned with the atmospheric domain) and ANAEE (concerned with ecosystems and biodiversity) are being manually ingested. Figure 29 shows the basic scenario:

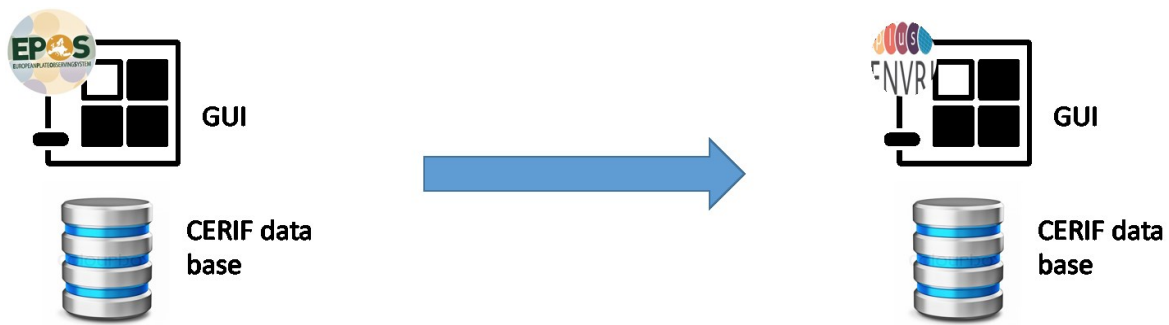


Figure 29 The CERIF catalogue solution provided by EPOS will be deployed more broadly in ENVRIplus.

6.1.2 Long term: automatically harvesting CERIF records from diverse ENVRI RI catalogues

In parallel to (a), an automated approach is also being prototyped by the University of Amsterdam within ENVRIplus. The basic idea is to take advantage of the metadata manager and metadata mappings developed in WP4 using the 3M environment, and to build upon that development by automating the manual pipeline currently used by FORTH to dynamically transform metadata records from ENVRI RI catalogues into a single CERIF database. Figure 30 shows the basic scenario:

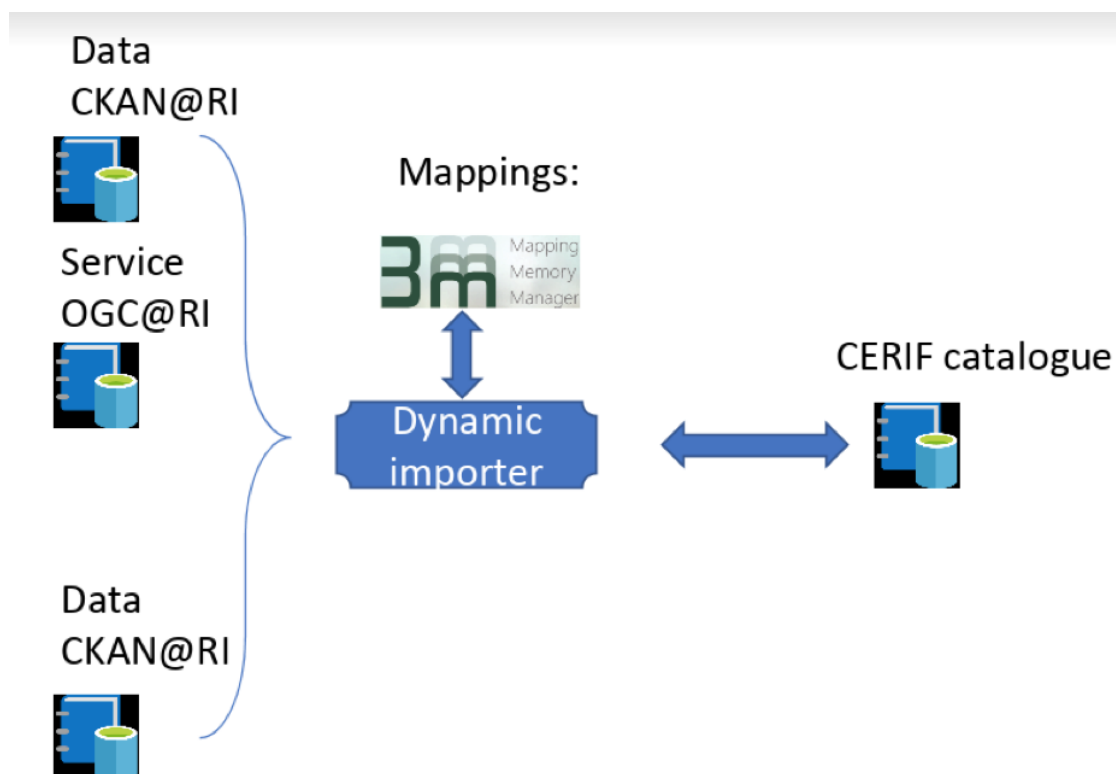


Figure 30 An automated pipeline for harvesting ENVRI RI catalogues into a single CERIF catalogue

Based on the automated pipeline shown in Figure 30, we have developed a service --named Metadata Catalogue Mapper (MetaCatMap)-- that implements a flexible metadata mapping pipeline using different metadata schemes to provide cross-RI metadata search and discovery.

Metadata Catalogue Mapper (MetaCatMap) consists of the following components:

- RESTCat: This is the REST frontend API that takes requests from users to perform the conversion from one standard to another. The parameters or the request are the source metadata catalogue url and the target mapping standard
- CatTask: This component extracts individual entries from the source metadata catalogue and creates a mapping task and adds it in a task queue
- CatMap: This component pulls a mapping task from the task queue, and queries the X3ML mapping framework for the appropriate mapping that matches the source and target standard
- X3ML Engine: This component performs the actual transformation of an entry from the source standard to the target standard

Figure 31 shows the overall architecture.

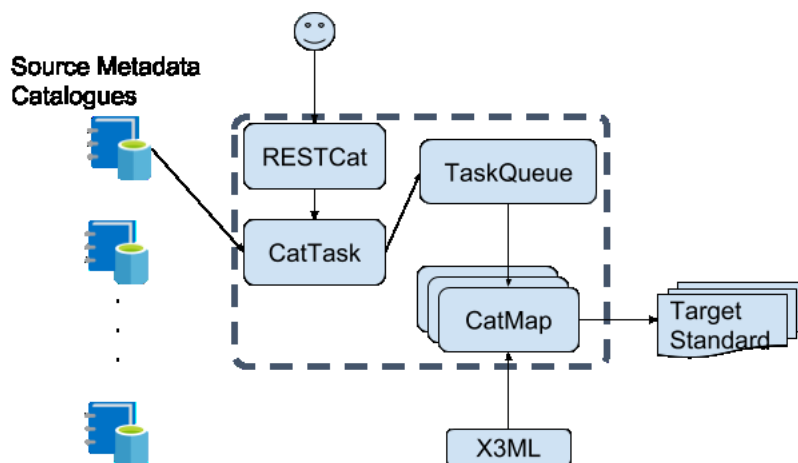


Figure 31 Architecture of the Metadata Catalogue Mapper (MetaCatMap)

6.2 Cross-RI data and service discovery

Following the construction of the joint catalogue based on CERIF, the semantic search capabilities being developed as part of the metadata service building block of the e-VRE will be in turn applied in ENVRIplus to enhance data and service discovery based on vocabulary linking activities now being conducted in ENVRIplus between RIs with similar but currently distinct vocabularies for describing environmental phenomena and observations. Since the metadata from ENVRIplus catalogues will be ingested and mapped onto CERIF, it is possible

to exploit the semantic classification layer of CERIF to take advantage of the linked vocabularies produced by ENVRIplus and so provide enhanced semantic search based on the metadata service developed by FORTH, as shown in Figure 32.

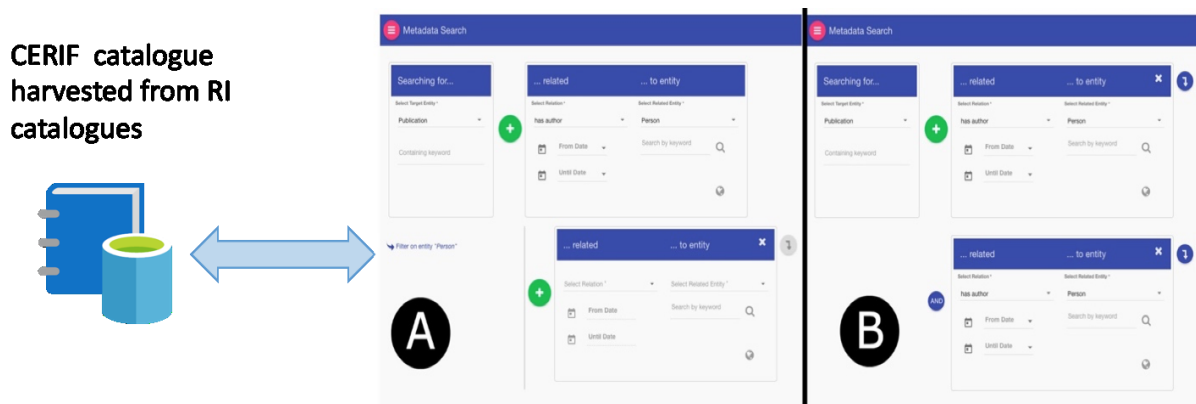


Figure 32 Using the semantic search support provided by the e-VRE to enhance cross-RI data and service discovery.

6.3 Cross-RI workflow composition

Using the semantic search capabilities of the metadata service and the workflow manager building block provided by the e-VRE, the ENVRIplus community is also able to enhance their cross-RI workflow composition capabilities. Figure 33 shows the basic scenario:

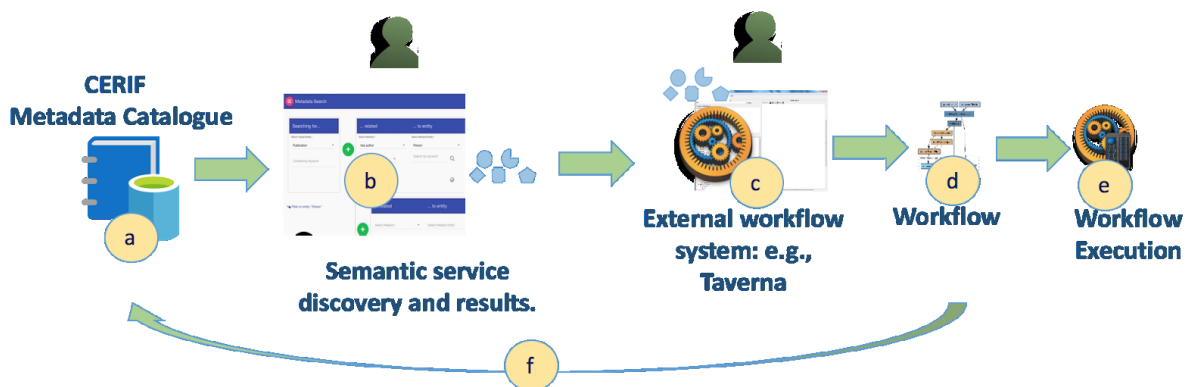


Figure 33 The workflow composition using eVRE building blocks.

The process can be broken down into six distinct steps:

- Ingesting metadata from data and service catalogues (in ENVRIplus) into a CERIF catalogue, as described earlier.
- Semantically searching data and services using the e-VRE metadata service with semantic search capabilities, likewise as described earlier.
- The workflow environment (in this case we use Taverna), will load the pre-selected services (identified in step **b**) into its own local service catalogue.
- Composing an executable workflow using Taverna.
- Executing the workflow using the Taverna engine.

- f. Storing the metadata of successful workflow compositions or executions back into the joint CERIF catalogue.

During the integration, the ENVRIplus team specifically compared two scenarios to show the added value of using semantic search, as shown in the figure below:

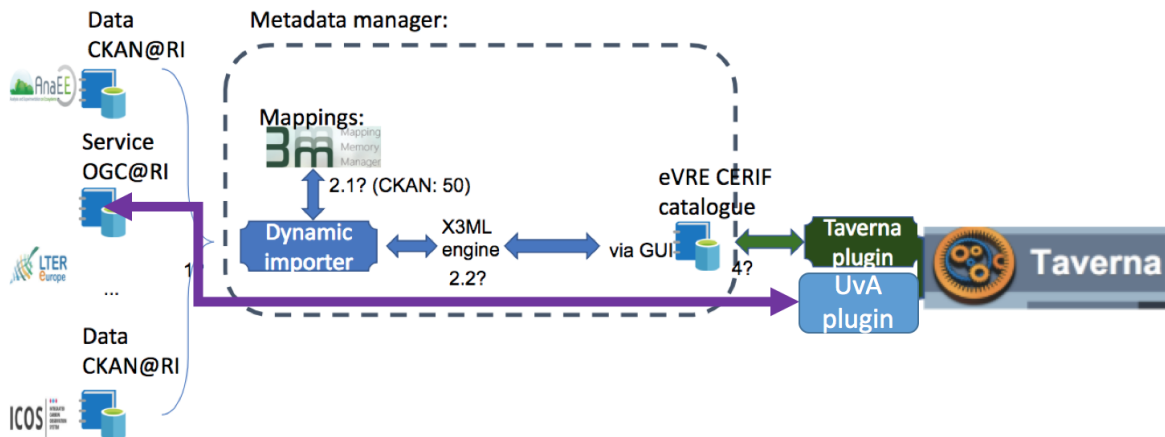


Figure 34 . Loading the RI service catalogue into Taverna directly (scenario a, indicated by the pink line via a UvA-developed plugin) or via CERIF and semantic search (scenario b, provided by the CERIF pipeline).

In the context of a single RI, scenario (a) can provide an effective, quick solution. In the case of workflows composed using services provided by multiple RIs however, it is judged that scenario (b) will provide better selection on data and services.

6.4 Cross-infrastructure workflow execution and provenance

Finally, the e-VRE building blocks have also been used to enhance workflow execution and provenance collection across multiple e-infrastructures. Figure 35 shows the basic scenario:

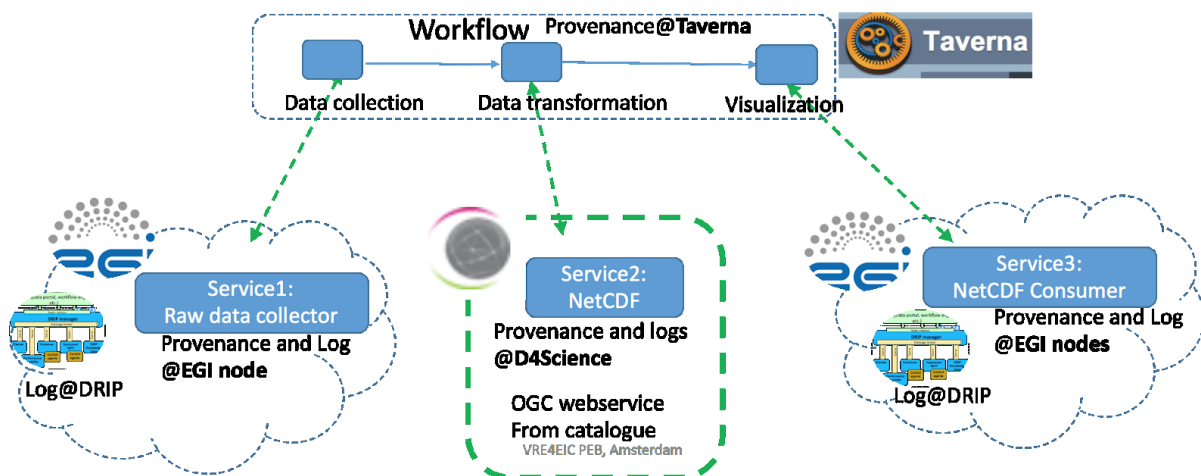


Figure 35 Cross-Infrastructure workflow execution and provenance

During the integration phase, the workflow manager (the e-VRE building block) will be used to perform execution, with services produced by the UvA team together with the ENVRIplus

provenance working group used to harmonize the distributed provenance information generated by the different distributed e-infrastructures at different levels on the technology stack, including workflow level, service level and infrastructure logs. A CERIF record will be created to link the different contexts of provenance data from different sources.

7 Conclusions

This deliverable has presented the four main software prototypical implementations of the VRE4EIC project:

1. the Canonical Reference Prototype (CRP), to be used as a model for future VREs, offering an implementation of the Reference Architecture of VRE based on the integration of state-of-the-art, open source technologies, such as the Unity system for AAAI and the Taverna Workflow system.
2. the eVRE Graphical User Interface, supporting the researcher in interacting with the CRP, and in browsing, querying and accessing the Metadata Catalogue.
3. the enhanced EPOS VRE, using the VRE4EIC technologies to realize two enhancements: (1) the move from a microservice centralized management, to a microservice choreography approach; and (2) the integration of the eVRE AAAI service into the EPOS architecture via the eVRENode service.
4. the enhanced ENVRIplus VRE, using the VRE4EIC technologies to realize two enhancements: (1) cross-RI data and service discovery, relying on the eVRE Metadata service; and (2) cross-RI workflow composition, relying on the semantic search capabilities of the metadata service and the workflow manager building block provided by the e-VRE.

The activity of the project continues on two main streams of work:

- the production of D3.5 “Final Architecture Design”, which will refine D3.1 with the findings on the Reference Architecture gained during the implementation of the building blocks and the CRP
- the development of demonstrators of the above four prototypes, showing the effective functionalities of the prototypes in realistic application scenarios.

8 References

[HOLL] Holl S. Automated Optimization Methods for Scientific Workflows in e-Science Infrastructures. Forschungszentrum Jülich; 2014.

[Newman] S. Newman, Building Microservices, O'Reilly Media. February 2015

[D33] https://www.vre4EIC.eu/images/Public_deliverables/D3.3_Building_Blocks.pdf

[RUSS] M. Russ, Going "Events-First" for Microservices with Event Storming and DDD, <http://www.russmiles.com/essais/going-events-first-for-microservices-with-event-storming-and-ddd#>

[D31] https://www.vre4eic.eu/images/Public_deliverables/D3.1_Architecture_Design.pdf

9 Annexes

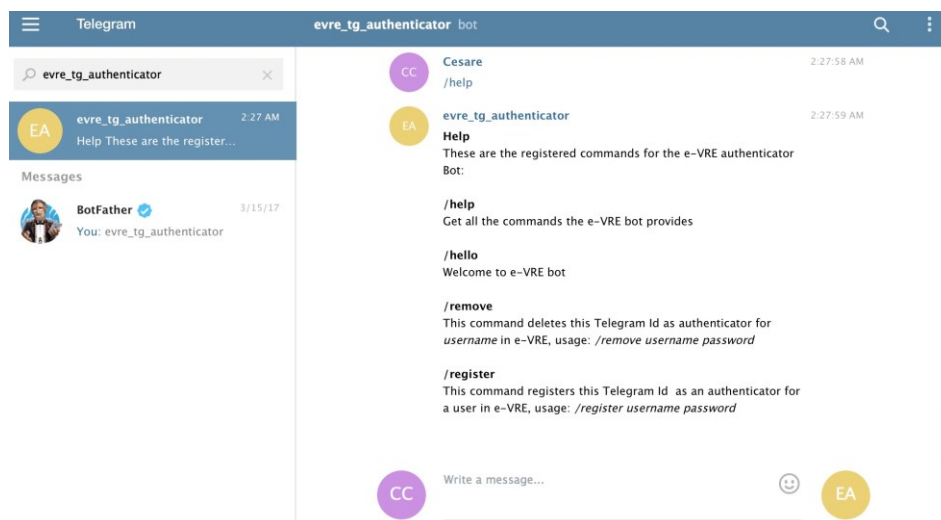
9.1 Using 2FA in e-VRE prototype

In order to implement the two factors authentication method (described in the AAAI section) we have developed a software module implementing a *Telegram Bot*: a software that enables the e-VRE to interact with the media platform Telegram.

Telegram Bots are special Telegram accounts that do not require a phone number to set up, the VRE4EIC bot has been registered with the id:

evre-tg_authenticator

A user that wishes to interact with e-VRE using Telegram, can add this account to his/her contact list and can start sending message.



The list of commands accepted by evre-tg_authenticator is:

- **/help**: Get all the commands the e-VRE bot provides
- **/hello**: Welcome to e-VRE bot
- **/register**: This command registers this Telegram Id as an authenticator for a user in e-VRE, usage: */register username password*
- **/remove**: This command deletes this Telegram Id as authenticator for username in e-VRE, usage: */remove username password*

These commands are forwarded by telegram to the e-VRE software module **TgAuthenticator**, developed from scratch using Java technology, that executes them.

When an e-VRE user sends the */register username password* command to `evre-tg_authenticator` bot, the TgAuthenticator sets up Telegram has the channel that will be used in two factors authentication for that user, this means that the second factor code will be sent to him/her via Telegram (note that Telegram channels are encrypted). The command */remove username password* instead disables the two factor authentication. In the future new commands will be added to this bot.

From the architectural point of view the TgAuthenticator is an independent software module, distributed as jar, that enables the e-VRE to be integrated with an external application (Telegram); in order to manage its life-cycle a basic version of the App manager has been implemented, it is able to start/stop TgAuthenticator and sends messages to other services when the status change.

9.2 VRE4EIC GUI implemented functionalities

Security	
F1	Login with two factors authentication;
F2	Role Based Access Control (RBAC) to regulate users' actions;
F3	User profile management;
F4	User Registration;
Data Presentation & Query Construction	
F5	Data classification with respect to VREs and RIs;
F6	The system is capable of assisting users to build and execute advanced queries;
F7	Provides a simple and user-friendly Graphical User Interface (GUI);
F8	Presents results in both tabular and geospatial form through suitable GUIs respectfully;
F9	Provides an interactive map to better assist end users into perceiving geographical data and construct geospatial data queries (allows setting geographical region or selecting instances);
F10	Provides SPARQL view of the constructed queries;

F11	Allows end users to store and load constructed queries, in such a way that all actions made through the GUI are preserved (and shown when loading one);
F12	Stored queries are associated to the user's profile and the potential of sharing them with other users or making them public (through the portal) will be possible (future work);
F13	Suitable statistics and numbers are provided when available.
F14	All input forms are validated before the submission of any entries, preventing end users from entering inappropriate input;
F15	Appropriate recommendations are available to the end users when necessary;
F16	The system prevents end users from constructing queries leading at no results, by showing only options and choices that make sense (options are dynamically adjusted);
F17	Appropriate tooltips or guidelines are available through the system for better usage assistance;
Data Import / Export	
F18	The platform allows data import from different file formats on existing named graphs or constructs a new one if necessary;
F19	Supports a variety of RDF based import file formats;
F20	Allows data export capabilities from final search results;
F21	Automatically acquires and uses any provenance information that is available from user's profile while importing.
Administration & Configuration	
F22	<p>A variety of configuration options for customizing the default behavior of the system is available through the UI; These are:</p> <ul style="list-style-type: none"> i. Setting limit on level and degree; i. Allowing usage of both regular expressions (OR / AND) or just the one chosen first; i. Selecting entities to be excluded from the list of available; r. Setting limit in the max number of instances selected per related entity;

	<ul style="list-style-type: none"> r. Setting the maximum number of results for which pins will instantly be displayed on the map when opening it; i. Setting the maximum number of results in a specified region for which pins will instantly be displayed on the map when opening it; i. Setting the maximum number of pins for which instances will be selected instead of the respective region drawn; i. Showing pins or not when drawing some region on the map; c. Always showing pins for selected instances or not
Robustness and Fault Tolerance	
F23	All possible errors are properly handled;
F23	System's functionality is preserved after recovering from any error potential error;
F24	Appropriate message reporting is provided to end users when necessary;

Table 1. Functionalities of the VRE4EIC Metadata Portal